

---

# Black Magic in Deep Learning

---

THESIS

submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

KANAV ANAND

Student id: 4712870

Email: k.anand@student.tudelft.nl

Thesis Committee:

Dr. Marcel Reinders, TU Delft (chair)

Dr. Willem-Paul Brinkman, TU Delft

Dr. Jan van Gemert, TU Delft (supervisor)

Dr. Marco Loog, TU Delft

Ir. Ziqi Wang, TU Delft



Pattern Research and Bioinformatics  
Faculty EEMCS, Delft University of Technology  
Delft, the Netherlands



# Preface

This thesis report has been written to fulfill the graduation requirements of a Master Degree in Data Science and Technology at the Delft University of Technology (TU Delft). The work done towards my thesis has helped me to extend my knowledge beyond the courses I have taken before the thesis work; challenging myself to be persistence and critical toward my work; as well as developing myself to be a better researcher and a better person.

I would like to give my gratitude to Jan van Gemert, who has allowed me to work under his supervision. Without his insightful input, this work wouldn't be able to be completed with sufficient quality. I would also like to thank Marco Loog, my 'unofficial' supervisor who has always been available with his valuable comments and out of the box ideas. My daily supervisor, Ziqi Wang, who was always there whenever I had doubts regarding my thesis work, giving on-point feedbacks as well as keeping the research direction on the track. Furthermore, I would like to thank the members of my thesis committee, Marcel Reinders and Willem-Paul Brinkman, for taking their time to read the report as well as attending my thesis assessment. I also wish to thank all the participants who volunteered and took part in the user study. Without their cooperation, I would not have been able to conduct this analysis. I would like to give my special thanks to my family for their consistent support, emotionally and financially, throughout this journey. Without their emotional and motivational support, I would not have been able to reach this point of completing my thesis.

I would also like to thank my 'mini' gang; Sashu– for *baby proofing* my report, giving up on your weekends for my thesis, consistently motivating me with your innocent lies and teaching me the compassion with your delicious vegan recipes. Dhruv– you unknowingly pushed me to be better and showed me how to be positive in the worst of times. I couldn't have asked for a better partner in situations like giving presentations, being surrounded by the board of examiners, the first time in the football locker room; Sanjeev– couldn't have asked for a better *daddy*. From picking up from the airport, giving us a place to stay and 'acting' like a noob on the field to make everyone else feel good about themselves. Dr. Nirmal– the focus and level of dedication is something I aspire to learn from you (probably the only thing). Tavishi– thanks for listening to my presentation. Presentations are less scarier for me after that day. Gouri– always being super chill and excited about new things. Rahul aka the chef for feeding all of us inf number of times and Bhatti for all the cocktails to drown my thesis sorrows. I might need more when my family reads this. Summi– the most innocent, smartest and honest 'guy'. You showed me how to be confident and real in any situation. Barbara – thanks for being so nice and all the pleasant conversations we had in last year. Lastly, Arka–

your constant yapping, mindless talking about food and charmless dance made my masters journey a challenge.

Kanav Anand  
Delft, August

# Understanding the role of humans in hyperparameter optimization

Jan Van Gemert<sup>1,5</sup>, Marco Loog<sup>2,5</sup>, Ziqi Wang<sup>3,5</sup>, and Kanav Anand<sup>4,5</sup>

<sup>1</sup>Assistant Professor, Pattern Recognition and BioInformatics, j.c.vangemert@tudelft.nl

<sup>2</sup>Associate Professor, Pattern Recognition and BioInformatics, M.Loog@tudelft.nl

<sup>3</sup>PhD, Pattern Recognition and BioInformatics, z.wang-8@tudelft.nl

<sup>4</sup>MSc student, K.anand@student.tudelft.nl

<sup>5</sup>Delft University of Technology

## Abstract

*Deep learning is proving to be a useful tool in solving problems from various domains. Despite a rich research activity leading to numerous interesting deep learning models, recent large scale studies have shown that with hyperparameter optimization it is hard to distinguish these models based on their final performance [15, 17]. The hyperparameter optimization has shown to improve the state of the art results on several occasions [10]. These results cast the doubts over the performance of these improved deep learning models and lead to the question whether the final performance of a deep learning model is dependent on the person performing the hyperparameter optimization task. A user study was conducted to evaluate the impact of human's prior experience in deep learning on the final performance of a deep learning model. 31 people with different levels of experience in deep learning were invited to perform a hyperparameter optimization task. The collected data was analyzed to find the relationship between human experience and the final performance of the deep learning model used for the user study. From the results, we observed that the final performance of the model varies with every participant, and a strong positive correlation exists between the participant's experience and the final performance. Our data suggest that an experienced participant finds better results using fewer resources.*

## 1. Introduction

Deep learning has made major leaps in solving problems from various domains. It has made major advances in the field of image recognition [5, 12], speech recognition [6, 20], analyzing particle accelerator data [3], bioinformatics [14, 16], sentimental analysis, question answer-

ing [2] etc. LeCun et al. [13] suggest that success of deep learning in a variety of domains can be attributed to the fact that it requires little engineering by hand and automatically learns representations and feature vectors of raw data used for the classification or detection.

Every deep learning algorithm is accompanied by a set of hyperparameters. These hyperparameters are external to the model and cannot be learned from the dataset explicitly. The optimal values for these set of hyperparameters lead to faster convergence, higher accuracy, and better results in general. These hyperparameters critically govern the performance of many learning algorithms [8], and simple optimization of these hyperparameters has been found to improve many of the existing state of the art results [10, 22]. However, finding the optimal values for these hyperparameters is a challenging task. The same deep learning algorithm will have different optimal hyperparameter configurations for different tasks [11]. Optimal configurations for one dataset do not necessarily translate to others [23]. This unpredictable behaviour of hyperparameters in hyperparameter optimization is informally referred to as the "black magic" in deep learning.

The hyperparameters are usually set by the human designer of the deep learning models. There have been studies that improved the existing state of the art results by performing just hyperparameter optimization [10]. The large scale experiments conducted in the field of natural language processing concluded that by using hyperparameter optimization techniques, vanilla Long short-term memory (LSTM) performs as good as recently designed state of the art methods [17]. Another study conducted to analyze different generative adversarial network (GAN) models suggested that if these models are trained and optimized sufficiently, there is no better model than vanilla GAN [15]. These results cast doubts over the performance of these improved deep learn-

ing models and lead to the question whether the final performance metric of the deep learning model varies when different people optimize the hyperparameters. It is essential as it would validate whether a model might lead to different final performance when trained by different individuals even in the absence of any source of variations except hyperparameters.

In research literature many deep learning models are published without the complete list of hyperparameters, and their values used to achieve the final performance [19]. This makes it difficult to reproduce the work presented in the literature and is implausible to replicate the stated final performance of the model. Thus, validating that final performance metric of the model varies based on the human performing the hyperparameter optimization will motivate authors to publish the complete list of hyperparameter values to make it reproducible.

It is widely assumed that hyperparameter optimization is a task reserved for experts [21]. The final performance of a deep learning model is *assumed* to be highly correlated with the human’s background knowledge tuning the hyperparameters. The validation of this assumption is one of the main goals of this research.

Thus, in this study, the relationship between human’s experience in deep learning and final performance of a deep learning model is investigated by letting people with different experience in deep learning perform hyperparameter optimization in a controlled setting. The data collected through this user study is used to answer the following research questions:

1. Research question 1: Is there any variance induced in the final performance of the deep learning model by hyperparameters?
2. Research Question 2: What is the relation between human’s experience in deep learning and the final performance of the deep learning model?
  - (a) Hypothesis 1: Humans have no impact on the final performance of the model.
  - (b) Hypothesis 2: Humans with more experience perform hyperparameter optimization task more efficiently.

In the following sections, we first describe the user study conducted to answer the research questions. We then present the analysis performed over the collected data of different individuals. Finally, we report the results and draw conclusions.

## 2. Approach

In this section, the methodology is discussed in detail. Specifically, it contains a description about the different

choices and setup of the user study. It is followed by an explanation of the data collected from the user study and the data analysis used to answer the research questions.

### 2.1. User study

#### 2.1.1 Participants Information

The main purpose of this user study is to evaluate the impact of human’s prior experience in deep learning on the final performance of a deep learning model. Therefore, it was important for this research that experience of participants is well distributed. In total, thirty-one participants completed the user study. The count of participants with their experience in deep learning is shown in Figure 1.

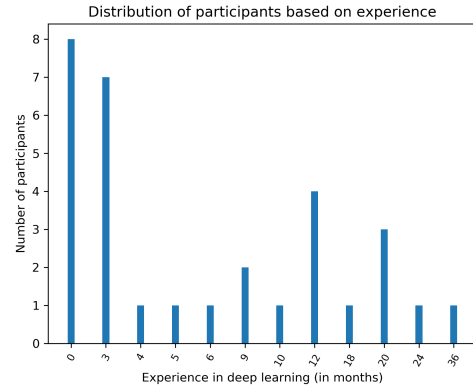


Figure 1: The distribution of experience in deep learning of participants that took part in the user study.

#### 2.1.2 Experiment Details

The experiment required participants to tune the hyperparameters of a deep learning model, *SqueezeNet* [9], for an object classification task with ten classes. The dataset used for the experiment, *imagenette*<sup>1</sup>, is a subset of *ImageNet*<sup>2</sup> [4] dataset. The dataset is comprised of 13000 training images, 500 images for validation set and 500 images for test set balanced among each class.

The Table 1 shows the list of hyperparameters that were used for the task. The values for the mandatory hyperparameters were required to start the training of the model, whereas if no value was provided for an optional hyperparameters then the default value, as suggested by the designers of the algorithm, was used. The task was to find the optimal set of hyperparameters maximizing the final performance metric, that is, the accuracy of the model on the test set. The participants were asked to submit the

<sup>1</sup>See <https://github.com/fastai/imagenette>

<sup>2</sup>See <http://www.image-net.org/>

values of these hyperparameters exposed by the model using a web interface. The participants were encouraged to add comments for each hyperparameter choice to submit their line of thoughts, however, it was not mandatory.

Upon the submission of hyperparameters, the model trained in the background using the submitted hyperparameters. While the model was training, the participant could view the intermediate batch loss and epoch loss in real time. The participant had an option to cancel training if the intermediate results do not look promising. As there was an upper time limit of 120 minutes, early stopping would enable the participant to try additional hyperparameter configurations. After training of the model is finished, accuracy on a validation set is provided to the participant. The experiment ends when the participant decides that the model has reached its maximum accuracy or time limit of the experiment is reached (120 minutes). The flow diagram of the user study is depicted in Figure 2.

Table 1: List of hyperparameters used for the user study.

Type	Hyperparameter	Default value
Mandatory	Epochs	-
	Batch size	-
	Loss function	-
	Optimizer	-
Optional	Learning rate	0.001
	Weight decay	0
	Momentum	0
	Rho	0.9
	Lambda	0.75
	Alpha	0.99
	Epsilon	0.00001
	Learning rate decay	0
	Initial accumulator value	0
	Beta1	0.9
	Beta2	0.999

### 2.1.3 Experiment Design Choices

The experimental design decisions were made with a rationale behind them. This section lists such important decisions and the rationale behind these decisions.

**Website Design:** One of the key requirements while designing this user study, was to replicate the hyperparameter optimization setting accurately. The web interface was designed to let participants submit their choice of hyperparameters, view their submission history with validation accuracy, and, most importantly, view the intermediate training results with an option for early stopping. The web interface provided the

freedom to include these functionalities and collect all the data for analysis.

**Choice of Task:** The choice of the task was crucial as it dictates the other decisions like the selection of model, dataset, and the hyperparameters involved. The object classification is one of the prominent and classical problems solved using deep learning. With a variety of available open-source dataset of ranging difficulties, it provided us with the freedom to experiment and choose from different models that fit into our user study requirements, especially time limit.

**Choice of Model:** The selection of the deep learning model was influenced by factors like size of the model, task in hand, total time taken for training and final performance of the model. Several prominent models like *LeNet* [24], *DenseNet* [7], and *Squeezenet* [9] were considered for the task. To find a balance between the training time and final performance of the model, *SqueezeNet* was chosen for this experiment. The *SqueezeNet* has fewer parameters and achieves a comparable level of accuracy to other deep learning models with much deeper architecture [9].

**Choice of Hyperparameters:** The epoch, batch size, loss function, and optimizer are the mandatory hyperparameter choices required to train the deep learning model. There is a plethora of options available for loss function and optimizer and, therefore, these options are restricted to the most commonly used. The mentioned restriction has been added since the upper time limit of the experiment is bounded, and it would be difficult for participants to test all possible combinations. There is a set of optional hyperparameters that is dependent on the choice of optimizer hyperparameter value. These optional hyperparameters come with default values, suggested by the designer of the optimizer algorithm.

**Choice of Dataset:** There were different choices that were considered before selecting *Imagenette*<sup>3</sup> [4] for the final experiment. This choice was influenced by the size and difficulty of the dataset. If the size is too big, it would mean more training time and in turn, more waiting time for the participants. Also, if the dataset is not challenging, it would be relatively easy to achieve a good final performance and might limit the variance in the final performance of the model.

**Masking Information:** The participants were provided with only partial information about the task and dataset. As the data is open source, a simple google

<sup>3</sup>See <https://github.com/fastai/imagenette>

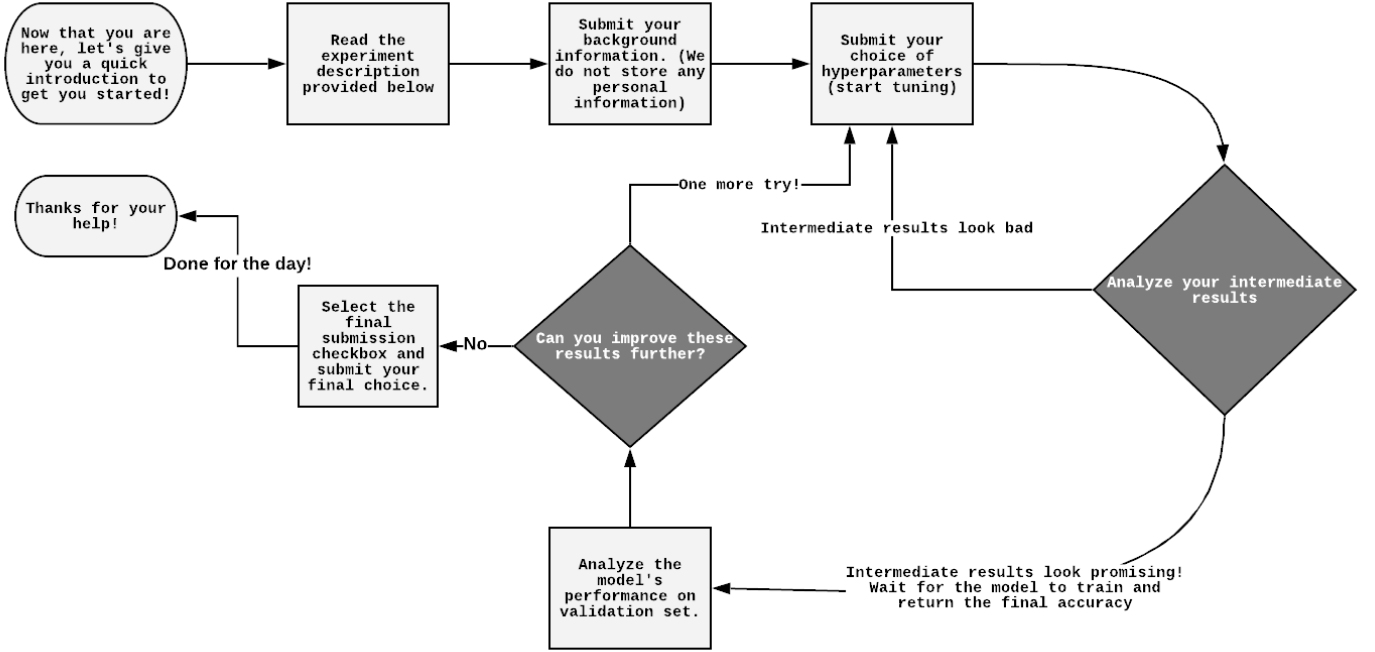


Figure 2: The flow diagram of the user study. The participant starts by entering their information. Next, submit the values for hyperparameters and evaluate intermediate training results. If the training is finished, the participant can decide whether to submit a new configuration for hyperparameter or end the experiment. It can be repeated until the time limit of 120 minutes is reached.

search could giveaway optimal hyperparameter configurations available for this dataset. The choice of model is majorly dependent on the type of task and dataset; therefore information about the model was also masked. They were informed about the number of classes and size of the dataset, including the partition information of Train/Val/Test.

## 2.2. Data Collection

There are two types of information stored for each participant, namely background information and experimental. The background information includes the number of months of experience in deep learning and highest education level at the time of the experiment. The experimental information contains all the hyperparameter combinations tried by the participant with the accuracy on the validation set, subjected to the model's completion of training. To indicate the end of the experiment, every participant was asked to submit their optimal choice of hyperparameter configuration. The final hyperparameter configuration for each participant is trained on training+validation set, and the final accuracy over the test set is recorded.

At the end of the user study, 463 different hyperparam-

eter combinations were collected from 31 participants. The final hyperparameter configuration submitted by each participant is used to train and test the model 10 times, and the average accuracy is reported as the final accuracy.

## 2.3. Data Analysis

For all the analysis, we have used  $p\text{-value} < 0.05$  to accept or reject the null hypothesis unless reported differently. That is, the null hypothesis is rejected if the p-value is less than the threshold (0.05); otherwise, it is accepted. As several experience categories have an uneven number of participants, the participants were divided into groups based on experience. The first group contains participants with 0 experience in deep learning, the second group with less than 9 months of experience and the third group with expertise greater than nine months. This allows us to analyze how accuracy is distributed under different groups and compare the distributions. Using the data collected through the user study, we aim to answer the following questions:

1. **Is there any variance induced in the final performance of the deep learning model by hyperparameters?**

Specifically, it is interesting to validate if there is any



variance induced by changing hyperparameter configurations for our user study setting. If there is no variance or minimal variance found, the impact of the participant’s experience on the final performance of the model cannot be quantified. We perform a random search over the restricted domain of each hyperparameter and store the final accuracy over the test set for each configuration. The variance<sup>4</sup> is reported over the final accuracy.

## 2. What is the relationship between human’s experience in deep learning and the final performance of the deep learning model?

After establishing that varying hyperparameter configurations impact and induce variance in the final performance of the deep learning model, we want to validate the hypothesis that different individual leads to the different final performance of the same model. The final performance of each participant is analyzed for a variance to accept or reject this hypothesis. Further, it was interesting to see how prior experience impacts the final performance of the deep learning model. We calculate the correlation between the user’s experience and the accuracy on the user’s final submission using *Spearman*<sup>5</sup> [1] rank-order correlation coefficient. Next, to find the quickest group, that is, the group that uses the minimum number of tries to achieve the highest accuracy we evaluate every participant’s hyperparameter configuration submissions with its accuracy on the validation set.

## 3. What are the different strategies followed by participants and how do they evolve with experience?

This is an exploratory question and is intended to provide a better understanding of the difference in the performance of the participants rather than conclusive evidence of which technique is better. We will make use of the data collected from the user study, specifically comments and values submitted by the users. The main goal is to understand the hyperparameter optimization techniques used by a participant in detail.

## 3. Results

This section explains and discusses data analysis in detail. Specifically, it describes the statistical analysis of the data collected from the user study and how it is used to answer the research questions.

### 3.1. Variance Due To Hyperparameters (RQ1)

The first task is to evaluate if there is any variance induced by hyperparameters in the final performance of the deep

<sup>4</sup>We use `numpy.var` method of python

<sup>5</sup>We have used `scipy.stats.spearmanr` method in python

learning model used for the user study. The data used for this experiment is collected by performing a random search over hyperparameter values and recording model’s accuracy. Figure 3 shows the variance in the task and model selected for the user study. The y-axis represents the number of occasions the model achieved the accuracy in between the range on the x-axis. The standard deviation, mean and variance were found to be 24.34%, 41.83% and 642.52 respectively.

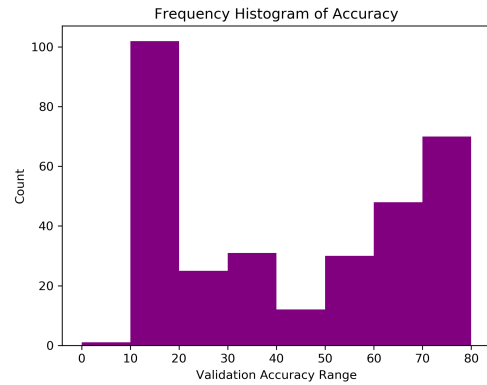


Figure 3: The frequency histogram for the accuracy collected over 320 different hyperparameter configurations using random search. The x-axis depicts the accuracy bins, that is, 0-10 is a bin containing all they hyperparameter configuration resulting in final accuracy in the range [0,10). The y-axis represents the count for that specific range. The maximum accuracy achieved is below 80% for the given model.

### 3.2. Humans and Hyperparameters (RQ2)

The final performance (accuracy) of the model was calculated on the test set using the hyperparameter configuration submitted as the final choice by each participant. In total, 31 participants completed the user study. The standard deviation, mean, and variance in the final accuracy were found to be 26.318%, 55.95 and 692.65 respectively. Figure 4 depicts the relationship between final accuracy and the experience in deep learning of each participant. As the experience increases, the final accuracy tends to increase. It is supported by the **strong positive** *Spearman* correlation of 0.6018 with p-value of 0.0004.

To analyze the relation between experience and accuracy further, we compared the distributions of Group 1, Group 2, and Group 3 using *Levene*<sup>6</sup> [18] test to validate if there is any variance in the accuracy between these groups divided by experience. The test values presented in the Table 2

<sup>6</sup>We used `scipy.stats.levene` method from python.

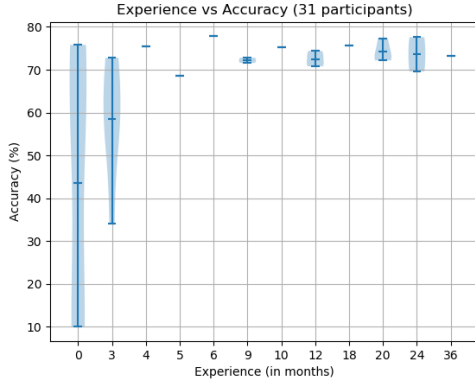


Figure 4: Distribution of final performance (accuracy) of all participants.

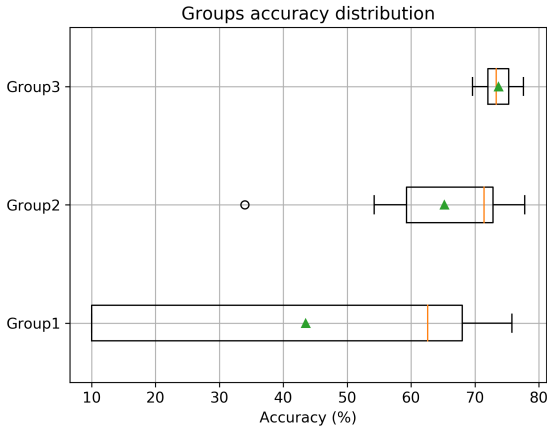


Figure 5: Distribution of final accuracy of each group. The green triangle represents the mean, while the yellow mark displays the median of the distribution.

show that each group has different distribution and most variation exists between Group 1 and Group 3 in line with the distribution shown in Figure 5.

Apart from observing a relation between final accuracy and experience of each participant, relating final accuracy to the experience of each individual, it was interesting to see how many different hyperparameter configurations each individual had to try before reaching a certain final accuracy. It shows the minimum resources used by participants to achieve the given accuracy. For the participant who did not achieve the given threshold accuracy, participant's total number of tries is used. Figure 6 depicts the distribution for Groups 1, 2, and 3 for different final accuracy as thresholds. Figure 7 shows the average trace of each group, that is, the average accuracy of each group after the specific number of

tries. The shaded region represents the standard error<sup>7</sup> of the distribution.

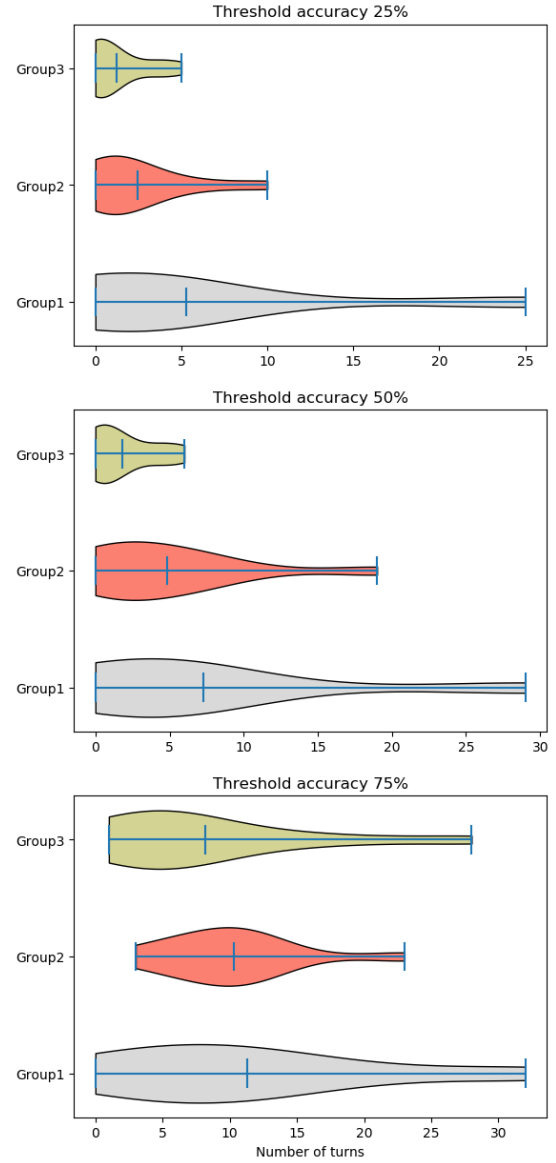


Figure 6: Number of different hyperparameter configurations required to achieve the threshold accuracy for different groups. The violin plot shown above depicts the probability density distribution of the number of turns taken by the participants in each group. The mean value of each group is marked for reference.

### 3.3. Difference in Strategies

In this section, we aim to find insights into the different strategies used by participants. First, we analyze the com-

<sup>7</sup>See [https://en.wikipedia.org/wiki/Standard\\_error](https://en.wikipedia.org/wiki/Standard_error)

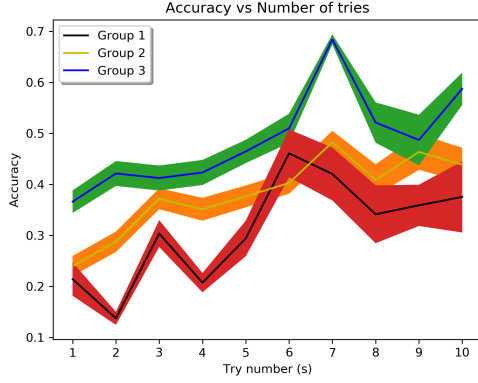


Figure 7: Average seen trace: it plots the average accuracy of the group observed after the specific number of tries. The shaded region represents the standard error<sup>8</sup> for the distribution.

Table 2: Levene test for comparing accuracy distribution between groups divided using experience.

Groups	Score	p-value
Group 1 vs Group 2	8.40	0.01
Group 1 vs Group 3	14.338	0.001
Group 2 vs Group 3	5.52	0.029

ments submitted by the participants to explain the choices made for each hyperparameter. We follow it up by the discussion on the use of default values that are suggested by the algorithm designers in the published literature.

### 3.3.1 Analyzing comments left by participants

Participants were encouraged to leave comments explaining the reasoning behind choosing a specific value of a hyperparameter. In a bid to gather maximum comments, we let users choose from predefined comments shown in Table 3. The Figure 8 shows the distribution of comments for each user and for each group(as explained in Section 3.2).

Figure 8a shows the comments percentage distribution for each experience category. The majority of comments for a participant with low experience is dominated by random guessing comments which indicate the random strategy adopted by these participants. The random guessing was found to be negatively correlated with the increasing experience. We used *Spearman* rank-order correlation, and the value was found to be  $-0.58$  with p-value  $0.0007$ . The group distribution of comments show a consistent decrease in random guessing and increase in using values from experience (green), as the amount of experience increases.

Table 3: Predefined comments used in user study and their corresponding numbers

Comment	Number
It is just a guess.	1
It is a suggested default value.	2
It is the value that has worked well for me in the past.	3
It is the value I learnt from previous submissions.	4
Other	5

### 3.3.2 Use of suggested default values

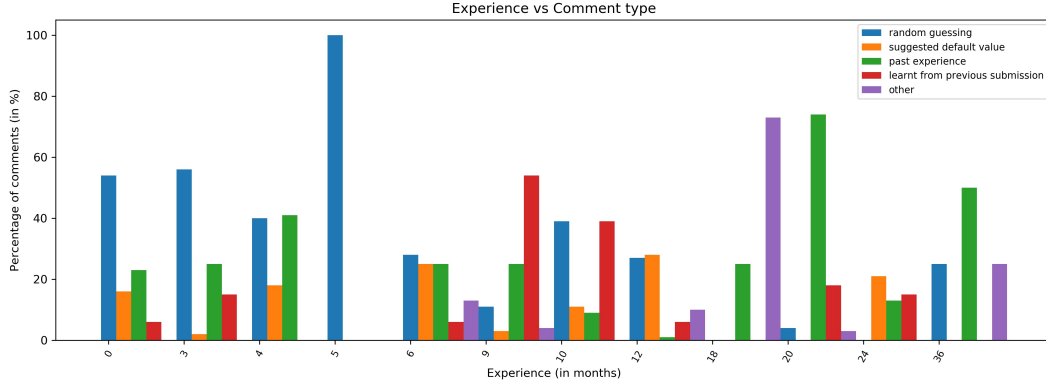
The hyperparameters are broadly divided into two categories - optional and mandatory in our user study, as shown in Table 1. These optional hyperparameters are conditional; that is, they belong to a specific optimizer algorithm and come into play only when the corresponding optimizer algorithm is selected. In the user study, participants were provided the default values for these conditional hyperparameters, which were suggested by the designer of optimizer algorithms. Figure 10 shows the number of participants in each group using these default values as the starting point. Except for four, every participant in Group 2 and Group 3 started with all suggested default values and built on it.

## 4. Discussion

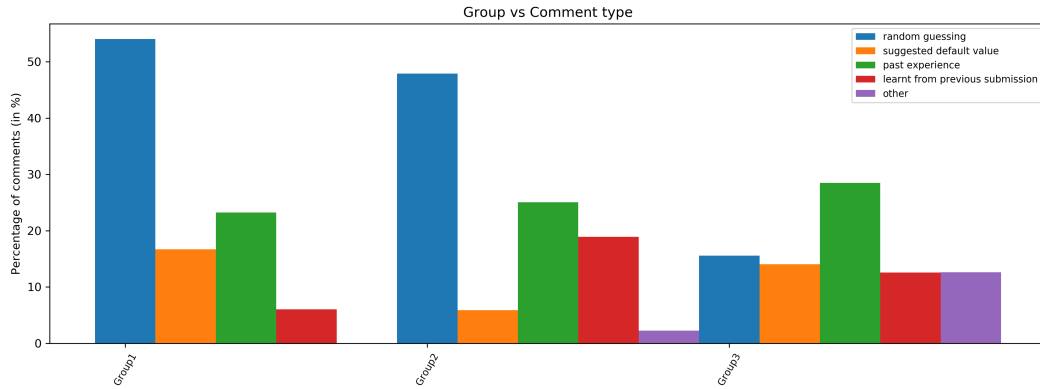
In this section, we will discuss the results based on the goals of this study set in the introduction (Section 1).

### I Humans do impact the final performance of the model

One of the problems discussed in the introduction was the difficulty in reproducing results of a published model without a complete list of hyperparameters and their values. Through this user study, we found people with different levels of experience tune a deep learning model, it performs differently. Every source of variation was eliminated by fixing the task, dataset, deep learning model, and the execution environment(random seed, GPUs used for execution) except the choice of hyperparameters. Figure 4 shows the variance in the final performance of the model. This suggests that final performance of the model is dependent on the human tuning it and it is difficult to reproduce the exact performance of a published model without the exact list of hyperparameter values used for training of the model.



(a) The comments analysis of participants grouped together based on experience. For multiple participants in the same experience category, average value is used.



(b) The comments analysis of each group showing the distribution of predefined comments.

Figure 8: Intermediate training results.

Submission #	Alpha.Value	Alpha.Comment.Value	Accuracy
1	0.75	2	68.6%
2	0.75	3	67.6%

Figure 9: An example depicting how a participant with no experience in deep learning confuses between the comment type 3 and comment type 4.

## II Role of humans in hyperparameter optimization

The main goal of this research is to understand the role of humans in hyperparameter optimization. We are interested in validating the hypothesis 2 that human with prior experience in deep learning is better suited for hyperparameter optimization task. Figure 4 shows a strong positive correlation between human's expertise level in deep learning and final performance of the model; that the final performance increases with an increase in experience of a participant. However, several experience categories have an

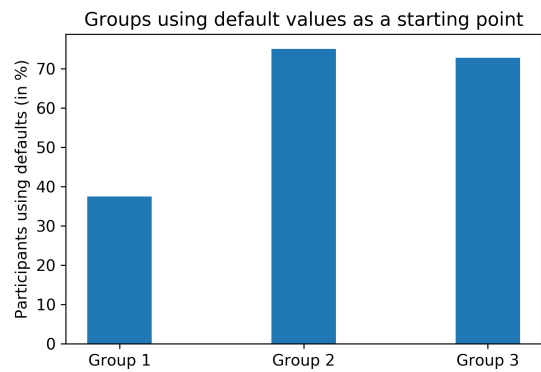


Figure 10: The number of participants in each group submitting their initial hyperparameter configuration using all default values suggested by the algorithm designer.

uneven number of participants. Therefore, the participants were divided into groups based on experience, and distributions were compared. The *Levene test* (Table 2) validated that the three distributions of final performance are different, which confirms that there is variance induced by prior experience in deep learning. The choice of *Levene test* was motivated by the fact that user’s experience data and the final performance are not normally distributed. The box plot in Figure 5 suggests that on average a participant with no expertise will achieve lower accuracy compared to experienced participants.

Figure 7 showing the average trace for different groups suggests the participants with higher experience achieve better accuracy in less number of tries. The distributions are shown in Figure 6 indicating similar findings for three different accuracy thresholds. Also, it can be seen that the average number of tries to reach 75% threshold is relatively closer. This is because not many participants reach that accuracy, and in that case, their total number of tries is considered, which are very similar.

### III Insights

The data collected from user study was analyzed to gather insights into the different strategies used by participants. The first comment analysis suggests that the people with fewer experience tend to use more random values for hyperparameters when compared to experienced participants. The experienced participant is more dependent on the values that have worked well for them in the past. From Figure 8b, we can see that Group 1, which contains participants with no experience in deep learning, has the second-highest comment percentage for prior experience. This means that 22% of values used by these participants were based on their prior experience in deep learning. An explanation for this high percentage could be the confusion between comments, namely ‘prior experience’ (3) and ‘learned from the previous submission’ (4). The ‘learnt from the last submission’ comment is meant to indicate the values determined from prior submissions and by analyzing the results during the experiment. One such example is shown in Figure 9. The participant was using the default value for alpha with appropriate comment (2) but for the next submission updated the comment to (3), which translates to ‘prior experience’ instead of ‘learning from the previous submission’.

Figure 10 shows the use of suggested default values by participants as the starting point. Majority of experienced participants tend to use these defaults more often compared to participants with no prior experience in deep learning. Although the use of defaults does not necessarily lead to optimal hyperparameter configuration, however, all those participants who started with defaults achieved final performance greater than 50%.

## 5. Conclusion and Recommendations

The main goal of this research was to understand the role of humans in hyperparameter optimization. We conducted a user study where people with different experience levels and knowledge in deep learning were invited to perform hyperparameter optimization in controlled settings.

Hyperparameter has been shown to have a significant impact on the final performance of the model. However, this impact varies with the settings used like dataset, model, task, etc. Thus, we first validated the effect of hyperparameters used for the user study (RQ1). The results showed that there is variance induced in the final performance of the model by varying the hyperparameter values. Next, to find the relationship between human and the hyperparameter optimization process, every participant was asked to submit their choice of optimal values of hyperparameter. The obtained final performance of each participant showed that the performance of a model is dependent on the individual performing hyperparameter optimization. A positive correlation between the experience in deep learning and the final performance of the model indicates that with an increase in knowledge, the final performance of the model gets better (RQ2).

Further analysis of the data collected from the user study showed that experienced participants are quicker to achieve better accuracy compared to less experienced ones. It indicates that people with more experience can achieve better results using fewer resources. The analysis of the comments submitted by participants provides us with insights into the strategy used by each participant. It suggests that random strategy is more prominent among people with lesser experience. The experienced participants tend to use values from their experience or preferably use a suggested default value instead of random guessing.

Concluding, we found a strong correlation between the performance of the model and the human performing the hyperparameter optimization. The people with more experience were found to be faster and better at hyperparameter optimization process.

Further, it provides proof that people differ in their strategies based on their prior knowledge. As people with lesser experience tend to follow a random approach, that is, they tend to choose values for their hyperparameter randomly while people with more experience use their prior knowledge to set the values for hyperparameters.

Apart from the task executed in this thesis, below is the list of recommendations that would add further value to the research.

1. There is always room for more data. Collecting more data and inviting more participants to perform the user study will make the result and conclusions even robust to outliers. It will provide a better insight into the

process of hyperparameter optimization and generalize our findings over a broader audience.

2. The inclusion of more participants could be done in a structured way. One of the limitations of the presented study is the different number of participants for each experience category. It makes it difficult to generalize the results for each experience category. Thus, while recruiting for more participants, the experience in deep learning could be divided equally.
3. Currently, only the experience in deep learning is used for each participant to account for the changes in the final accuracy of the model. This could be expanded to include more background information of each participant like age, gender, ethnicity, education level, expertise in a specific field of deep learning, etc. Also, adding more dimensions to the background information for each participant would require more data to make any concrete conclusions.
4. This study currently focuses on a single task, single model, and unique dataset. Thus, it can be argued that the findings of this study could not be generalized for the entire field of deep learning. Hence, the next step could be to extend this study further by including multiple tasks, different models like GAN, RNN, Deep Reinforcement Learning, and various datasets.

## References

- [1] *Spearman Rank Correlation Coefficient*, pages 502–505. Springer New York, New York, NY, 2008.
- [2] A. Bordes, S. Chopra, and J. Weston. Question Answering with Subgraph Embeddings. *arXiv e-prints*, page arXiv:1406.3676, Jun 2014.
- [3] T. Ciodaro, D. Deva, J. M. de Seixas, and D. Damazio. On-line particle detection with neural networks based on topological calorimetry information. *Journal of Physics: Conference Series*, 368:012030, jun 2012.
- [4] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.
- [5] N. L. L. Y. Farabet C. Couprie C. Learning hierarchical features for scene labeling. In *IEEE Trans Pattern Anal Mach Intell*. 2013 Aug;35(8):1915-29. IEEE, 2013.
- [6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov 2012.
- [7] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely Connected Convolutional Networks. *arXiv e-prints*, page arXiv:1608.06993, Aug 2016.
- [8] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 754–762, Beijing, China, 22–24 Jun 2014. PMLR.
- [9] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv e-prints*, page arXiv:1602.07360, Feb 2016.
- [10] R. P. A. Jasper Snoek, Hugo Larochelle. Practical bayesian optimization of machine learning algorithms. *Bartlett et al. [8]*, pp. 29602968, 2012.
- [11] R. Kohavi and G. H. John. Automatic parameter selection by minimizing estimated error. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, ICML’95, pages 304–312, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
- [12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436 EP –, May 2015.
- [14] M. K. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics (Oxford, England)*, 30(12):i121–i129, Jun 2014. 24931975[pmid].
- [15] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs Created Equal? A Large-Scale Study. *arXiv e-prints*, page arXiv:1711.10337, Nov 2017.
- [16] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik. Deep neural nets as a method for quantitative structure-activity relationships. *Journal of Chemical Information and Modeling*, 55(2):263–274, Feb 2015.
- [17] G. Melis, C. Dyer, and P. Blunsom. On the State of the Art of Evaluation in Neural Language Models. *arXiv e-prints*, page arXiv:1707.05589, Jul 2017.
- [18] I. Olkin. Contributions to probability and statistics; essays in honor of Harold Hotelling. *Stanford, Calif., Stanford University Press, 1960.*, 1960.
- [19] Y. Rao and J. Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, Dec 2016.
- [20] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran. Improvements to deep convolutional neural networks for LVCSR. *arXiv e-prints*, page arXiv:1309.1501, Sep 2013.
- [21] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, Jan 2016.
- [22] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *arXiv e-prints*, page arXiv:1206.2944, Jun 2012.

- [23] J. N. van Rijn and F. Hutter. Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 2367–2376, New York, NY, USA, 2018. ACM.
- [24] Y. B. P. H. Yann LeCun, Lon Bottou. Gradient-based learning applied to document recognition. *arXiv e-prints*, Aug 1998.

# Contents

<b>List of Abbreviations</b>	xvi
<b>1 Background Information</b>	<b>1</b>
1.1 Hyperparameter Importance	1
1.1.1 Beating The State of the Art	2
1.1.2 Evaluation of Neural Language Models	2
1.1.3 Are All GANs Equal?	3
1.2 AutoML	4
1.2.1 Introduction	4
1.2.2 Grid Search	5
1.2.3 Random Search	5
1.2.4 Evolutionary Strategies	6
1.2.5 Bayesian Optimization	7
1.2.6 Multi-Fidelity	8
1.2.7 NAS - Neural Architecture Search	10
<b>2 User Study Design</b>	<b>12</b>
2.1 The Experiment	12
2.1.1 Background Information	13
2.1.2 Hyperparameters	14



2.1.3	Intermediate Visualizations . . . . .	18
2.1.4	Dataset . . . . .	19
2.1.5	Model Used . . . . .	20
2.2	Design Motivation . . . . .	22
2.2.1	Web Platform . . . . .	23
2.2.2	Task . . . . .	23
2.2.3	Hyperparameters . . . . .	23
2.2.4	Dataset . . . . .	24
2.2.5	Deep Learning Model . . . . .	24
2.2.6	Hiding Certain Information From Participants . . . . .	24
2.3	Participant Information . . . . .	25
2.3.1	Recruiting Participants . . . . .	25
2.3.2	Participant Data . . . . .	26
<b>3</b>	<b>Experiments and Results</b>	<b>28</b>
3.1	Experimental Setup . . . . .	28
3.2	Variance Due To Hyperparameter . . . . .	28
3.3	Final Performance and Humans . . . . .	29
3.3.1	Accuracy vs Experience . . . . .	31
3.3.2	Group Analysis . . . . .	34
3.4	Strategies of the Experiment . . . . .	35
3.4.1	Analysis of Comments . . . . .	36
3.4.2	Use of suggested default values . . . . .	39
	<b>Bibliography</b>	<b>40</b>
<b>A</b>	<b>Statistical methods used for analysis</b>	<b>45</b>

A.1	Spearman rank-order correlation method . . . . .	45
A.2	Levene’s test . . . . .	45
<b>B</b>	<b>Final submission of each participant</b>	<b>47</b>
<b>C</b>	<b>User study screenshots</b>	<b>49</b>
<b>D</b>	<b>Poster and Flyer</b>	<b>53</b>

# List of Abbreviations

Abbreviation	Description
AutoML	Automate Machine Learning
BOHB	Bayesian Optimisation HyperBand
CMA-ES	Covariance-Matrix Adaptation Evolution Strategy
CNN	Convolutional Neural Network
FID	Frchet Inception Distance
GPU	Graphics Processing Unit
GP	Gaussian Process
GAN	Generative adversarial network
GB	Giga byte
HPO	Hyperparameter Optimization
KDE	Kernel Density Estimator
LSTM	Long short-termmemory
MCMC	Markov Chain Monte Carlo
NAS	Neural Architecture Search
RNN	Recurrent Neural Network
SVM	Support Vector Machine
SGD	Stochastic Gradient Descent

# List of Figures

1.1	Exploring of different hyperparameter values in the grid and random search strategies. Figure reproduced from [1]. . . . .	6
1.2	Outline of the successive halving algorithm. After each set of iterations, the number of configurations is reduced to half, and only the best one survives to finish the complete training. Figure reproduced from [2]. . . . .	10
2.1	Background Information collected for each participant. . . . .	13
2.2	Interface to submit the values for hyperparameter. Next, to every hyperparameter field, a comment field is given to enter the reasoning behind the value provided by the participant. . . . .	13
2.3	The number of training examples in the dataset and the batch size together decide the number of iterations required to complete one epoch. The figure is reproduced from an online blog post. . . . .	15
2.4	Intermediate training results. . . . .	19
2.5	Imagenette Dataset example image from each class. . . . .	20
2.6	The two components, squeeze and expand layer, forms the fire module in the <i>SqueezeNet</i> . The squeeze layer reduces the number of channels for expanding layer, therefore, effectively reducing the number of trainable parameters. . . . .	21
2.7	<i>SqueezeNet</i> model's building blocks and the architecture. The figure is reproduced from <i>SqueezeNet</i> published literature [3] . . . . .	22
2.8	The accuracy plot for <i>LeNet</i> over <i>Imagenette</i> dataset. The x-axis represent the hyperparameter configuration number found using random search and y-axis is the final accuracy for that configuration. The maximum accuracy achieved is 56.8% while the minimum being 10% . . . . .	25
3.1	An overview of analysis done to answer RQ1. . . . .	29

3.2	The frequency histogram for the accuracy collected over 320 different hyperparameter configurations using random search. The x-axis depicts the accuracy bins, that is, 0-10 is a bin containing all they hyperparameter configuration resulting in final accuracy in the range [0,10). The y-axis represents the count for that specific range. The maximum accuracy achieved is below 80% for the given model. . . . .	30
3.3	An overview of the analysis done for RQ2. . . . .	31
3.4	The distribution of experience in deep learning of participants that took part in the user study. . . . .	32
3.5	The distribution of final accuracy of participants that took part in the user study. .	32
3.6	Distribution of final performance (accuracy) of each participant. . . . .	33
3.7	Distribution of final accuracy of each group. The green triangle represent the mean while yellow mark displays the median of the distribution. . . . .	35
3.8	Average seen trace: it plots the average accuracy of the group observed after the specific number of tries. The shaded region represents the standard error for the distribution. . . . .	36
3.9	Number of different hyperparameter configurations required to achieve the mentioned threshold accuracy by available groups. The mean value of each group is marked for each distribution. . . . .	37
3.10	Intermediate training results. . . . .	38
3.11	An example depicting how a participant with no experience in deep learning confuses between the comment type 3 and comment type 4. . . . .	38
3.12	The number of participants in each group submitting their initial hyperparameter configuration using all default values suggested by the algorithm designer. . . . .	39
C.1	The terms and condition webpage used for the user study. It clearly states all expectations and requirements for participants. . . . .	49
C.2	The background information page is shown above. It asks the participants to enter their relevant experience in the field of deep learning and their highest completed education level. Once the information is submitted by the user, anonymous user id is generated to identify every unique participant. . . . .	50
C.3	The experiment details page provides the detailed information of the user study. It starts with the flow diagram of the user study and follows it up with the explanation of every hyperparameter used for the experiment. . . . .	51

C.4	The hyperparameter submission form is used by the participant to submit the values for hyperparameters and their corresponding comments. Once a hyperparameter configuration is submitted, the user can view intermediate results and early stop the training of the model. . . . .	52
D.1	Poster used to display on the screens of majority of faculty buildings at TU Delft. .	53
D.2	Flyer used to recruit people for user study. . . . .	54

# List of Tables

2.1	List of hyperparameters used for the user study. . . . .	14
2.2	Comments available for participants. . . . .	27
3.1	The hardware information of four machines used for the user study. The software configuration of all these machines are same. GB refers to Giga Byte . . . . .	29
3.2	The result table showing the evaluation metric for different machines used for user study. . . . .	29
3.3	The <i>D'Agostino and Pearson's</i> normality test results. The null hypothesis is accepted as the p-value for both data distribution is $<0.05$ . . . . .	31
3.4	List of group and participants. . . . .	34
3.5	Levene test for comparing accuracy distribution between groups divided using experience. . . . .	34
A.1	Interpreting different values of $r_s$ . . . . .	45
B.1	The final submission of each participant. List of abbreviations used for the headings are listed in Table B.2 . . . . .	47
B.2	List of abbreviations used to display the final submission table. . . . .	48

# Chapter 1

## Background Information

In this chapter, we first provide background information to make the readers familiar with the concepts involved in our research topics. Secondly, we offer a brief literature survey of research discussing their approach, merits, and limitations that are related to our work. To the best of our abilities, we could not find any research that is directly related to our goal - analyzing the role of the human in deep learning hyperparameter optimization. However, there is much-related research that has been conducted to identify the importance of hyperparameter optimization in the field of deep learning.

In the following sections, we first describe the work done to explore the importance and role of hyperparameter optimization in the area of deep learning. We then present the automated machine learning (AutoML) methods published to eliminate the need of human from the process of hyperparameter optimization. Many AutoML strategies are inspired by the way humans train a deep learning model. For example, Bayesian methods with early stopping are motivated by the way humans analyze the loss and validation loss data during the hyperparameter optimization process.

### 1.1 Hyperparameter Importance

There has been recent surge in publications trying to emphasis on the need for efficient hyperparameters optimization and questioning the performance of recent complex deep learning models. It has been shown several times that when more budget and time allocated for hyperparameters optimization, there is very little or no performance difference between vanilla and complex models [4–6]. The conclusions drawn from this research is two folds: First, there is a need to allocate more resources for hyperparameter optimization process as it highly impacts the model performance. In addition to the need for resources, there should be more emphasis on the research done to mitigate the black magic in the hyperparameter optimization process and develop a better understanding of the entire process. Secondly, it questions the need for researching new complex models as with efficient hyperparameter optimization; it is difficult to differentiate between vanilla and complex models.



The following subsections explain these publications in more detail, providing a brief introduction of the research, presenting their key result, merits, and limitations.

### 1.1.1 Beating The State of the Art

This paper refers to the process of hyperparameter optimization as a "black art" and provides an automatic method to find optimal hyperparameter settings for any given learning algorithm. The method is based on the Bayesian approach using Gaussian Process (GP) to model the distribution of the final performance of the learning algorithm. The distribution expects the set of hyperparameter values as input and provides the expected performance of the learning algorithm. In addition to maximizing the final performance of the model, this AutoML method tries to minimize the time and resources by following a greedy approach. It exploits the hyperparameter space with a high probability of impacting the final performance.

The paper uses this GP based Bayesian approach to find the optimal hyperparameters for a 3-layered convolution network used for object classification on the dataset CIFAR-10<sup>1</sup>. An expert for comparisons performs the same task. The Bayesian approach found the optimal hyperparameters that improved state of the art (at the time of publishing) by over 3% and about 3.98% better than the expert. The only limitation of the algorithm is the number of training iterations required by the AutoML approach. It demands a bigger budget and more resources when compared to human optimization approach.

### 1.1.2 Evaluation of Neural Language Models

This represents a large scale study conducted to evaluate the various state of the art neural language models. All these neural language models are based on vanilla recurrent neural network (RNN) that aims to improve upon the "shortcomings" of the vanilla model [6]. This research re-evaluates the Long short-term memory (LSTM) models by using large scale black-box hyperparameter optimization over the same task to eliminate any source of variation.

The models were evaluated on different datasets with varying size and difficulty. With large computation budget allocated, 1000 run for each model, the vanilla LSTMs outperformed other recently introduced complex models. The LSTM architecture achieved new state of the art results on Penn Treebank<sup>2</sup> and Wikitext-2<sup>3</sup> corpora. The research concludes that with the huge amount of computation, the new models can be easily outperformed or at least achieve similar performance compared to the vanilla model. There lies a trade-off between the amount of computation and the results reported. Thus, there is an apparent need for solutions to make model evaluation cheaper

---

<sup>1</sup>See <https://www.cs.toronto.edu/~kriz/cifar.html>

<sup>2</sup>See [https://www.ling.upenn.edu/courses/Fall\\_2003/ling001/penn\\_treebank\\_pos.html](https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html)

<sup>3</sup>See [https://github.com/pytorch/examples/tree/master/word\\_language\\_model/data/wikitext-2](https://github.com/pytorch/examples/tree/master/word_language_model/data/wikitext-2)

by reducing the hyperparameters or model’s performance sensitivity to it and researching better hyperparameter optimization strategies.

### 1.1.3 Are All GANs Equal?

The purpose of the study was to evaluate different generative adversarial network (GAN) algorithms and determine the possible differences in the overall performance among these algorithms. Various GAN algorithms have been introduced recently<sup>4</sup>, and a majority of them have managed to achieve notable results. The paper designed the experiments and metrics to test prominent GAN algorithms on neutral grounds. The results show that if trained for long enough, all these algorithms achieve similar performance level and no modified GAN algorithms consistently outperform the original GAN [4]. The main goal of this study is to provide a rich comparison of major GAN algorithms and identify which algorithms perform better (or worse) than others. This study also focuses on presenting a methodology to compare different GAN algorithms fairly and without any bias.

The authors use FID (Frechet Inception Distance), precision, recall, and F1 score to evaluate the models. Metrics like log-likelihood and inception score are discussed. However, the research claims FID is the best fit and robust to handle the task in hand, supported by empirical evidence. The paper approximates the values of precision, recall, and F1 score for each model using data manifolds. The study uses an intuitive way to calculate these metrics efficiently. A new dataset is created (Toy dataset) with a known probability distribution. The squared Euclidean distance is used to find the resemblance between generated and real samples. If all different types of images are generated, the recall is high. Moreover, precision is high when the distance between generated images and training images is low.

Further, this study evaluates each model using two primary experimental setups. The difference lies in the choice of hyperparameters. The first experiment, referred as wide one-shot setup, identifies 100 samples of hyperparameters using random search, whereas the other setup, termed as narrow two-shot setup, uses 50 examples of hyper-parameters selected manually using the results of wide one-shot setup over the single dataset, FASHION-MNIST<sup>5</sup>. Other design decisions explained in the study include the choice of datasets (4 datasets from simple to medium complexity), selection of the architecture, INFO-GAN [7], random seed to make the initial weights in architecture random and computational budget. The budget is represented in terms of hyperparameters samples used to find the best fit for the model.

For wide range setup, high variance in FID metric was found for each model. There was no single model which was found to be significantly stable than others. While for narrow range setup, the results shown are quite mixed. There are some models which are found to be more sensitive to hyperparameters than others. Overall, FID values were reported much lower for wide range hyper-parameter search. The study computes precision, recall, and F1 score over the toy dataset. It was found that NS-GAN outperforms other models in terms of F1 scores. The study shows the

---

<sup>4</sup>See the ever-growing list of all GANs <https://github.com/hindupuravinash/the-gan-zoo>

<sup>5</sup>See <https://www.kaggle.com/zalando-research/fashionmnist>

impact of increasing the computational budget. With the increase in budget, the minimum FID of the model decreases while precision and recall increase for the majority of the models except for BEGAN [8].

The results show that no model dominates other models. It was found that by increasing the computation budget(or performing more hyperparameter optimization) can improve the performance of the model. Conversely, with a low budget, it is difficult to make any significant inference of the model’s performance. Further, the claims of some models outperforming the Original GAN model are not supported by the found empirical evidence. The study concludes that future research in GAN comparison should be done on neutral grounds and more focus is required to understand and implement the hyperparameter optimization techniques better.

## 1.2 AutoML

The advent of new powerful and computationally expensive machine learning algorithms have fueled the interest of many researchers in the field of hyperparameter optimization (HPO). This chapter begins with a general introduction of the HPO problem, followed by an overview of the different approaches that have been used for performing HPO.

### 1.2.1 Introduction

Every learning algorithm has a set of hyperparameters, and the task to obtain the set of optimal values for these hyperparameters is called HPO. The optimal set of hyperparameters leads to faster convergence, higher accuracy, and better results in general. These parameters govern the performance of any learning algorithm to a huge extent, and simple optimization of these hyperparameters has been found to improve many of the existing state of art results [5].

Let  $\chi$  be a learning algorithm with  $N$  hyperparameters. The domain of each hyperparameter is denoted by  $\Lambda$ , where the domain of  $n$ -th hyperparameter is represented by  $\Lambda_n$ , and the combined configuration space as  $\Lambda = \Lambda_1 \times \Lambda_2 \times \dots \times \Lambda_n$ . Thus, the learning algorithm using a vector of hyperparameters instantiated to  $\lambda$  can be expressed as  $\chi_\lambda$ .

There are different ways to express the domain of a hyperparameter based on its type. It can be real-values, integer-type, Boolean, or categories. Furthermore, the existence of certain hyperparameters can be dependent on a conditionality, that is, a hyperparameter only comes into existence only if another hyperparameter is set to some specific value. For example, while using different machine learning algorithms as a categorical hyperparameter, many other algorithm-specific hyperparameters become active only if the corresponding algorithm is being used. Thus, the problem of HPO can be formally presented as:

$$\lambda^* = \arg, \min_{\lambda \in \Lambda} F(L, \chi_\lambda, D_{train}, D_{validation}) \quad (1.1)$$

where  $F(L, \chi_\lambda, D_{train}, D_{validation})$  is the final performance metric generated by the learning algorithm  $\chi$  when trained using a set of hyperparameters initialised using vector  $\lambda \in \Lambda$ .

### 1.2.2 Grid Search

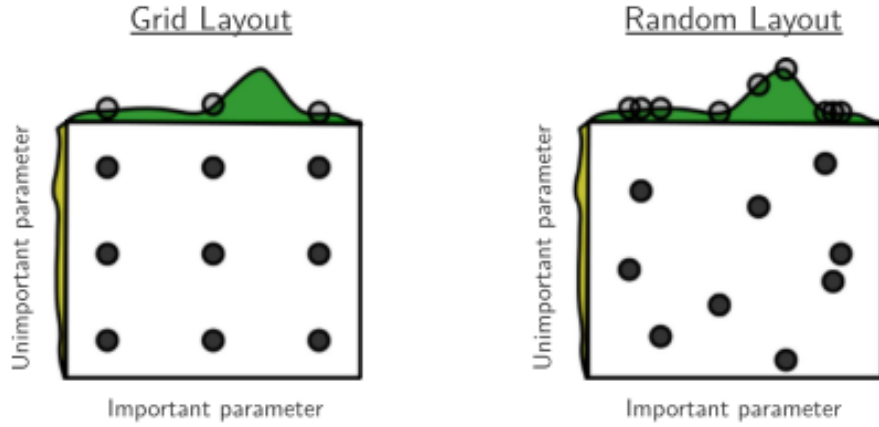
The first automated algorithm to search over the configurable space  $\lambda$  is done by exploring every possible set of values. The user bounds the domain for each hyperparameter, and grid search algorithm evaluates the possible Cartesian product over these hyperparameter sets [9].

Although grid search explores every possible set of values and guarantees optimal results, it also suffers from the curse of dimensionality [10]. As the number of possible set combinations increases, the possible set of combinations increases exponentially. In real-life scenarios, HPO is bounded by the budget, and thus the number of values that can be evaluated for each of the  $N$  hyperparameters are limited to  $Budget^{\frac{1}{N}}$  [11], where  $N$  is the total number of hyperparameters. In the majority of the cases, it is seen that some hyperparameters are more important for the learning algorithm than other [1],[12].

### 1.2.3 Random Search

The major drawbacks of grid search motivated Random search [1]. Random search generates different possible sets of hyperparameters by randomly choosing the value of each hyperparameter. It tends to work better than grid search when some hyperparameters are more important than others as it explores more values for each hyperparameter under a certain budget, see Fig. 1.1.

The random search is widely used as the baseline for many other HPO approaches as it does not make any assumptions about the underlying machine learning algorithm and is expected to achieve nearly optimal performance if given enough resources. Although random search gives positive results, it is still non-adaptive, that is, it does not take into account the result of the experiment before deciding the new set of hyperparameter values. Also. It tends to fail on highly sequential optimization tasks, performing worse than grid search [1].



**Figure (1.1)** Exploring of different hyperparameter values in the grid and random search strategies. Figure reproduced from [1].

#### 1.2.4 Evolutionary Strategies

The evolutionary strategies generate a set of candidate solutions of the hyperparameter values. Based on the performance of the model obtained for the solutions, the strategy updates the model distribution and generates a new set of candidate solutions with the motive that this new set will produce better performance metric for the model [13].

These two strategies differ in the way they use the results to generate a new set of candidates. The evolution strategy samples the candidate sets of solutions from a normal distribution and based on the evaluation results; this normal distribution is updated by replacing the new mean with the best performing candidate from the prior sample. The next set of candidate solutions is sampled around the new mean. Another strategy - genetic algorithm - considers only top 10% of the candidate samples to make decisions for the next candidate generation. It randomly selects two possible sets from this sample and generates new samples. Due to the greedy nature of these strategies, it is likely to get stuck at a local optimum. These strategies do not explore the area because the sample is always limited in space due to constant variance. If the variance is increased, it becomes difficult for these strategies to converge at one solution. CMA-ES solves this problem by adaptively increasing or decreasing the search space based on the result obtained for candidate solutions. It samples candidate solution set from a multivariate normal distribution and updates its  $\mu$ ,  $\sigma$ , and  $\Sigma$  based on the results obtained.

Evolutionary strategies are simple to implement and execute in parallel [14]. However, as the number of parameters increases, it suffers a setback in terms of performance due to covariance calculation.

### 1.2.5 Bayesian Optimization

Bayesian optimization was the first autotuned network to win the competition against human experts [15]. It gained interest by achieving state of the art results for image classification using neural networks [16],[17]. Bayesian optimization can be described as an iterative algorithm that aims to predict the distribution of the models final performance metric with respect to hyperparameters. It is comprised of two major components: a surrogate model and an acquisition function. The results obtained through model evaluations are used to fit the surrogate model to a distribution. The acquisition function uses this distribution to sample new configuration set of hyperparameters that maximizes the final performance metric of the model. It is designed to find the trade-off between exploitation and exploration; that is, one of the main objectives of the acquisition function is to balance the sampling of points from the region of high certainty and unknown regions. This is necessary to avoid getting stuck at a local optimum. Many different models have been published, varying based on the choice of surrogate models to acquisition functions used. A few of the prominent approaches are presented in the following section below.

#### Gaussian Process

The Gaussian Process (GP) has been proven to provide a strong prior over functions, and it has been widely utilized for properties like smoothness and their measurement of uncertainty estimates. The GP works on a principle that any finite set of observations with a Gaussian prior induces a multivariate Gaussian. Initial prior can be used to plug in the expert knowledge about the learning model distribution. The GP is defined using a mean and a covariance function. The mean is assumed constant or centered around 0. Thus, the quality of a GP is defined by its covariance function. The covariance function is represented as a Kernel where the common choice is squared exponential kernel. However, this choice of the kernel was found to give very smooth distributions and do not apply to the majority of practical problems [5].

One of the challenges of Bayesian optimization is to achieve parallelism. Since the algorithm tries to learn from each model evaluation, simple batch parallelism loses important information. The Markov Chain Monte Carlo (MCMC) estimates of acquisition functions using data from evaluated results and models being evaluated. The calculation of covariance function has been shown to scale cubically with the number of data points. This limits the number of configurations that can be evaluated under the given budget. Many new approaches have been suggested, for instance, using cylindrical and additive Kernels, evaluating model only on a subset of data, use of random embeddings to trade off space for speed [18].

#### Random Forests

Another majorly used surrogate model used for HPO are Random Forests. These are simple machine learning models generally used for classification and regression tasks and have been proven to

be highly suitable for categorical data [19]. It is considered to be the ensemble of regression trees where each life contains the learning algorithm final performance metric. It starts with randomly sampling configurations for evaluation, and using these evaluations a random forest is generated. Every node is split when minimum number of hyperparameters are considered and the number of configuration evaluations used at that node is greater than  $n_{min}$ . These are the parameters set by the user. The Regression forests predictive mean and variance is used to calculate the distribution of learning model function distribution.

Random forests boost high parallelism, easy to scale to high dimensions and work well with high categorical parameters, unlike GP. The scaling for fitting the surrogate model and predicting the variance is in the order of  $\text{Log}(n)$  and  $n \cdot \text{Log}(n)$ , respectively, compared to  $n^2$  and  $n^3$  for GP [20].

### Tree-Structured Parzen Estimators

Contrary to Gaussian process, where the model distribution is predicted using  $P(Y/x)$  where  $x$  is a given set of hyperparameters and  $Y$  is the model final performance metric; Tree Parzen takes the opposite approach. It replaces the prior distribution with  $l(x)$  and  $g(x)$ , both represents non parametric densities of the points. The  $l(x)$  represents the density of points where the corresponding model performance was observed to be lower than  $y^*$ , whereas  $g(x)$  represents the points with performance greater than  $y^*$ . The  $y^*$  is chosen to be some quantile of the observed performance metric and keeps on updating with each new evaluated result. These observed densities are used to maximize the expected improvement to find new candidates for evaluation. It is solely dependent on the ratio  $\frac{g(x)}{l(x)}$ , which means for maximum improvement it is ideal to draw points from  $l(x)$  density region. The configuration set with the greatest expected improvement is returned [20]. It has been widely used to optimize parameters for deep neural networks for image classification, speech recognition, and neural language modelling [21].

### 1.2.6 Multi-Fidelity

All approaches presented in the above sections need the model to complete the training and learn from the results obtained. They treat the execution of the model as a black box, and thus, with the onset of complex architecture and huge datasets, training exceeds several hours or even days [22]. One novel way to search for optimal hyperparameters is to train the model on a smaller subset of training data, by using a subset of features or reduce the size of images et al. Another approach includes predicting the model distribution based on the data collected using fewer iterations and early stopping the training of models [23].

The Bayesian optimization does improve the performance of human experts but at the cost of more resources or budget. One of the main reasons is that, unlike humans, Bayesian optimization algorithms do not observe the intermediate results in their learning. The prediction of the learning curve of the models performance, based on a few model iterations, can be used to predict the



final performance of a model. This predicted final performance value is compared with the current best value to decide if the training should be continued. It records the performance of the model at regular intervals and a linear combination of 11 parametric functions to generate the model distribution for the given set of hyperparameters. MCMC inference is used to predict the final performance value of the model. This approach was used to predict the optimal combination of 52 hyperparameters of image classification on Cifar-10 using ten fully connected layers. It managed to speed up the HPO process by the factor of 2 when compared to Bayesian optimization (SMAC, TPE).

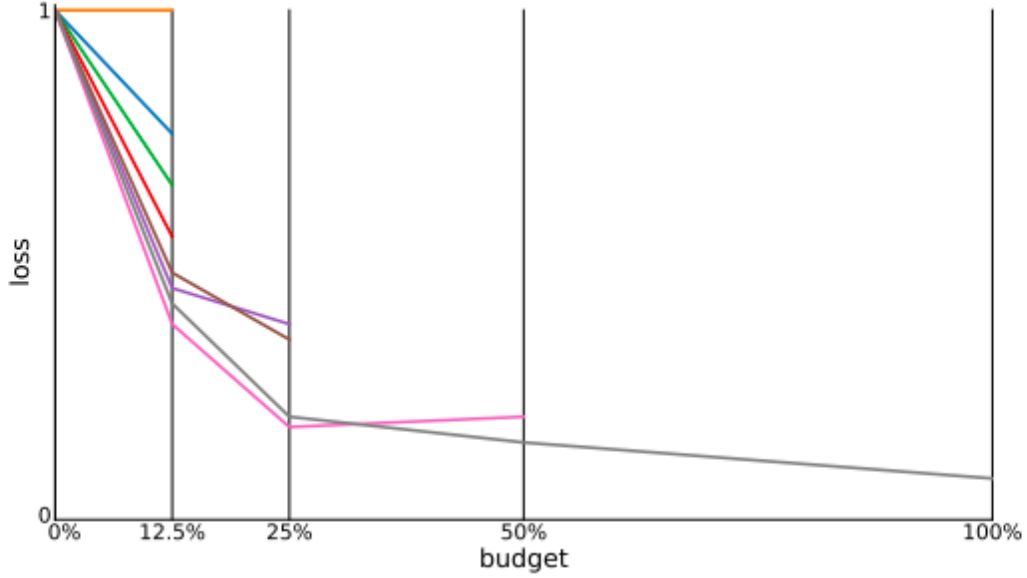
Another similar approach, Freeze-Thaw Bayesian optimization, explores  $N$  possible hyperparameter configurations at a time for  $m$  intervals and uses these results to refit the Gaussian process. It eliminates worst performing models and replaces them with new hyperparameter configurations generated using Gaussian process and acquisition function [24].

The use of bandit-arm strategy for HPO improved upon the existing strategies in terms of the budget. The successive halving algorithm shifts the focus on the budget available for optimization of hyperparameters [25]. It randomly samples the initial set of configurations of hyperparameters that will be used to evaluate the model. Initially, all configurations were allowed fixed set of iterations  $r$  and based on the obtained performance results, worst performing half of these configurations is dropped. Thus, in the end, only the best performing configuration is returned. The complete algorithm for a set of 8 different combinations of hyperparameters is shown in Fig. 1.2. The shortcoming of this approach is the need of trade-off between the number of configurations and budget available. The user can choose to assign more budget to a small number of configurations, which creates the problem of wasting resources on poor configurations by training them for a longer period, or small budget for more number of configurations that causes premature stopping of promising configurations.

Hyperband was introduced to combat this problem of finding an appropriate balance between the number of configurations and the budget. Another motivation to introduce Hyperband was to explore more algorithms based on a random search. The Bayesian optimization was found to surpass random search but only by little margins [17]. The random search is conceptually simple, easy to implement and is embarrassingly parallel which makes it a viable option to pursue [14].

Hyperband selects multiple configurations at random and divides them into multiple batches of the successive halving algorithm. Each batch returns their best performing configuration, and out of these, the top-performing algorithm is returned. It showed improvements over random search and black box Bayesian optimization algorithms and took only constant time more than vanilla random search when used for HPO of deep neural networks [26]. Another interesting algorithm, BOHB, aims at combining the best of Hyperbands early stopping approach with prominent Bayesian approach to learn the new configurations from the past results rather than randomly sampling over the configuration space. It uses the collected dataset including the partial run of each model and uses it to generate a new batch of configurations. It uses a similar approach to Tree Parzen estimator but rather than using a multiple single-dimensional KDE; it uses single multidimensional KDE. To fit this kernel at least  $D+1$  evaluations are required, where  $D$  is the number of hyperparameters being tuned. The BOHB tries to balance the exploitation and exploration while finding the next set of hyperparameters by choosing  $\rho$  fraction of total samples randomly. It outperforms all the state-of-





**Figure (1.2)** Outline of the successive halving algorithm. After each set of iterations, the number of configurations is reduced to half, and only the best one survives to finish the complete training. Figure reproduced from [2].

the-art methods of HPO for tasks like SVM classification, deep neural networks, and reinforcement learning [18].

### 1.2.7 NAS - Neural Architecture Search

The HPO takes place after the model architecture is designed and implemented. However, designing and implementing complete architecture requires expert knowledge and time. Thus, neural architecture search aims to develop a complete architecture and find the optimal set of hyperparameters using just the validation set. One of the first variants of NAS used re-enforcement learning with gradient-based optimization to find the optimal combination of architecture and its hyperparameters [27]. It uses an RNN based controller that updates the architecture and choice of hyperparameters using a performance metric of the model created. The performance metric of the model is used as a reward signal to update the current model to maximize the reward. Although architecture choices are limited to CNNs and RNNs, NAS provides a wide range of choices ranging from enabling skip connections, pooling, local contrast, et al.

The results show that NAS was able to build a model for CIFAR-10, which achieved 0.09% better performance and 1.05 times faster than state-of-the-art. During this experiment, NAS came up with staggering 12,800 intermediate architectures, and 800 GPUs were used concurrently for training purposes. It becomes clear that one of the major shortcomings of NAS using re-enforcement learning is an extensive requirement of computational resources.



# Chapter 2

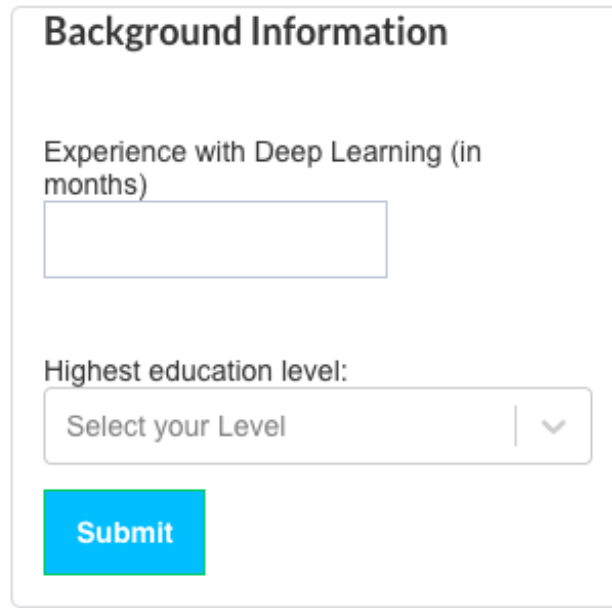
## User Study Design

This chapter explains the user study designed to collect the participant’s data used to answer the research questions. It attempts to explain the design choices considered and provides the reasoning behind the decision. It starts with a general introduction and flow of the experiment, followed by a brief explanation of each major component in the research.

### 2.1 The Experiment

The experiment aimed to replicate a hyperparameter tuning scenario in the best possible way. To facilitate hyperparameter optimization for the participant with no experience with deep learning or computer science, we decided to make a simple user interface. The user interface is divided into three significant components - participant’s background information, interface to submit the values for hyperparameters, and live graphs to view intermediate training results.

The participant submits their number of experience in deep learning and highest current education level using participant’s background information. After submitting their background information, the participant selects the values of hyperparameters and submit it using the hyperparameter interface. The participants were motivated to add comments for each hyperparameter to understand the reasoning behind the selected value. Once the hyperparameters are submitted, the model starts training in the background, and intermediate visualization is made available for the participant to analyze the impact of selected hyperparameter values. All three components are shown in Figures [2.1](#), [2.2](#) and [2.4](#), respectively.



**Background Information**

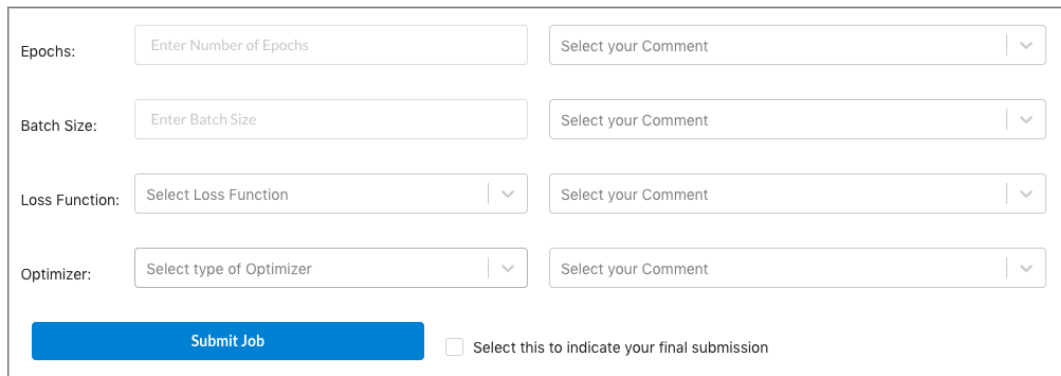
Experience with Deep Learning (in months)

Highest education level:

Select your Level | v

Submit

**Figure (2.1)** Background Information collected for each participant.



Epochs: 

Select your Comment | v

Batch Size: 

Select your Comment | v

Loss Function: 

Select Loss Function | v

Select your Comment | v

Optimizer: 

Select type of Optimizer | v

Select your Comment | v

Submit Job

☐ Select this to indicate your final submission

**Figure (2.2)** Interface to submit the values for hyperparameter. Next, to every hyperparameter field, a comment field is given to enter the reasoning behind the value provided by the participant.

### 2.1.1 Background Information

Every participant was asked to submit their background information - experience (in months) in the field of deep learning and highest education level. The expertise in areas related to deep learning like machine learning, artificial intelligence, etc. is excluded. This study aims to find the relation between the participant's experience and the final performance of the deep learning model. The highest education level information could help us understand the variance in the final performance, if any, within the participants with similar experience level. The participants were not asked to submit any other personal information that could be used to trace the final results of the experiment

back to a specific participant.

### 2.1.2 Hyperparameters

The hyperparameters are the key component of our user study. These hyperparameters are divided into two sets of categories - mandatory and non-mandatory hyperparameters. The complete list of hyperparameters is shown in Table 2.1. The mandatory hyperparameters are required for the model to start training. The non-mandatory list of hyperparameters are conditional hyperparameters; that is, it is dependent on the choice of optimizer hyperparameter. These are the hyperparameters that control the behavior of the optimizer algorithm and, if the participants provide no value, the default values suggested by the designer of these algorithms are chosen. Below we provide brief information about each hyperparameter used in this study.

**Table (2.1)** List of hyperparameters used for the user study.

Type	Hyperparameter	Default value
Mandatory	Epochs	-
	Batch size	-
	Loss function	-
	Optimizer	-
Optional	Learning rate	0.001
	Weight decay	0
	Momentum	0
	Rho	0.9
	Lambda	0.75
	Alpha	0.99
	Epsilon	0.00001
	Learning rate decay	0
	Initial accumulator value	0
	Beta1	0.9
	Beta2	0.999

**Epoch:** The epoch controls the number of times the optimizer algorithm iterates over the entire data set. It is defined before training of the model, and, one epoch is finished when an entire dataset has completed both forward as well as backward iteration exactly once. The minimum value for this field is 1 as there is at least one epoch required to go through the complete dataset. Generally, a model requires more than one complete iteration over the dataset to learn the problem distribution better. The Figure 2.3 shows the differences between epoch, batch, and iteration<sup>1</sup>.

---

<sup>1</sup>See <https://www.quora.com/What-is-an-epoch-in-deep-learning>

**Batch Size:** When dealing with large datasets, one epoch is too big to feed to the algorithm at once. So, one epoch is divided into multiple batches. A batch is the total number of training examples used for one forward and backward pass by the optimizer algorithm. Figure 2.3 explains the relation between epoch, batch size, and iterations. The minimum value required for batch size is one as the optimizer algorithm needs at least one training example to learn the correct value of weights.



**Figure (2.3)** The number of training examples in the dataset and the batch size together decide the number of iterations required to complete one epoch. The figure is reproduced from an online blog post.

**Loss Function:** Every deep learning model learns using a loss function. The loss function is a mathematical way to measure how wrong the predictions are of your model. It is a method of evaluating how well the model has learned the given problem. If the prediction deviates too much from truth results, the loss function is usually high. The participants were given different choices of the loss function to choose from, and each option is explained below:

#### 1. Cross entropy<sup>2</sup>:

Cross entropy loss measures the performance of a classification model whose output is a probability value for each class depicting the confidence for each category. Cross-entropy loss increases as the predicted probability diverge from the actual label. A perfect prediction would have a log loss of 0 as it would have value 1 for the correct class and zero for the rest. The equation 2.1 shows the calculation of cross-entropy loss for a single training example. The cross-entropy is represented as the combination of softmax function<sup>3</sup> and negative likelihood. The  $x$  contains the output from the last layer of the model for each class, and  $class$  is the index of the correct label.

$$loss(x, class) = -\log \left( \frac{\exp[x[class]]}{\sum_j \exp(x[j])} \right) \quad (2.1)$$

<sup>2</sup>We use `torch.nn.CrossEntropyLoss` method from python

<sup>3</sup>See [https://en.wikipedia.org/wiki/Softmax\\_function](https://en.wikipedia.org/wiki/Softmax_function)

## 2. L1 loss<sup>4</sup>:

It is also known as the L1-norm loss function. The L1 loss is minimizing the sum of the absolute differences between the target value and the estimated values. It calculates the distance between the predicted values and the true labels for each training example. The equation 2.2 depicts the L1 loss of a single training example where  $x_n$  is the output value of the model for class  $n$  and  $y_n$  is the correct value.

$$loss(x, y) = L = \{l_1, l_2 \dots l_n\}^T, l_n = |x_n - y_n| \quad (2.2)$$

## 3. Mean squared loss<sup>5</sup>:

It is also known as the L2-norm loss function. The mean squared loss calculates the difference between the model's predictions and the ground truth, square it and average it across the complete dataset. The equation 2.3 shows the mean squared loss calculation for one training example  $x$  with the truth value  $y$ .

$$loss(x, y) = L = \{l_1, l_2 \dots l_n\}^T, l_n = (x_n - y_n)^2 \quad (2.3)$$

## 4. Negative likelihood<sup>6</sup>:

The negative log-likelihood loss is high if the model puts low confidence at the correct class, and it is low when the model puts high confidence in the right class. The equation 2.4 represents the calculation of negative likelihood for a single training example. The  $x$  contains the output values for each class passed through a softmax function.

$$loss(x, class) = -\log(x[class]) \quad (2.4)$$

**Optimizer:** During the training process, the model updates its weight and parameters to minimize the loss function and make predictions more accurately. The optimizer connects the loss function with the model's parameters and weight by updating the model in response to the output of loss function. The optimizer tries to shape and mold the model into its most accurate form. The loss function guides the optimizer by guiding the model to update the weights in the right direction.

The optimizer algorithms have been used extensively in the field of deep learning [28–30]. Below are the several prominent optimizer algorithms that are widely used in the field of deep learning and have been chosen as the different options for the choice of optimizers.

---

<sup>4</sup>We use `torch.nn.functional.l1_loss` method from python

<sup>5</sup>We use `torch.nn.MSELoss` method from python

<sup>6</sup>We use `torch.nn.NLLLoss` method from python

### 1. Stochastic Gradient Descent (SGD):

The SGD is a variant of the most popular optimization technique used in deep learning, Gradient Descent [31]. The gradient descent is fast, robust, and easily adaptable to multiple domains. Gradients are partial derivatives of the weights in the model and are a measure of change. It calculates the impact of small change in the weights of the model on the loss function. Guided by the loss function, the weights are updated to move in the right direction. These two steps are repeated until the loss function is as low as possible. The gradient descent has been used across all types of Machine Learning and other math problems as an optimization technique.

SGD used gradient descent to optimize the model, but instead of using every training example in the dataset to update the weights, it uses single or multiple examples as a batch for each pass. The convergence time increases for gradient descent as the size of dataset increases, and thus, SGD leads to faster convergence.

### 2. Averaged SGD:

The averaged SGD, as proposed in [32], is a variant of SGD that aims to reduce the impact of noise present in the training examples. The complete procedure of optimization is similar to SGD, but instead of updating individual weights, the averaged SGD takes the mean of weights as the final solution.

### 3. Adagrad:

The learning rate is found to be a very difficult hyperparameter to set [33]. If it is set too small, then the parameter update will be slow, and there are high chances of getting stuck into a local minimum. Otherwise, if it is set too high, there are chances of never settling at the optimum loss. The learning rate is usually set constant for all parameters in different dimensions, which make things worse as each parameter has different sensitivity on each dimension.

The Adagrad is one of the first well-known algorithms to mitigate this issue by adaptively scaling the learning rate in each dimension. The learning rate is scaled using the accumulated squared gradient collected in each dimension over time. Thus, the dimension with less frequent updates receives high learning rates. Although, due to the accumulation of gradient over time, the Adagrad suffers from learning rate decay problem. The learning rate becomes close to 0 in a few dimensions, which stops the learning in the model. As this algorithm decreases the learning faster for more frequent parameters, and relatively slower for a non-frequent parameter, it is more suited for sparse data.

### 4. RMSProp:



RMSProp was never published in the scientific literature but was taught in one of the lectures in the course Neural Networks for machine learning<sup>7</sup>. The RMSProp also belongs to an adaptive learning rate family and uses the moving average to adapt the learning rate based on the squared gradient received for each parameter. As the name root mean square suggests, RMSProp uses the square root of the mean squared gradient value to modify updates for each parameter. Similar to Adagrad, the updates with time becomes smaller as the squared gradient increases. It is ideal for a convex problem with one minimum but for a non-convex problem it creates a problem as there are high chances of getting stuck in local minima.

## 5. Adadelta:

The Adadelta and RMSProp were stemmed out at the same time, trying to fix the shortcomings of Adagrad. The Adadelta tries to minimize the learning rate decay problem by limiting the past gradient values considered to adapt the learning rate [34]. It tries to consider the last  $W$  values of the past gradient to calculate the moving average and combined with the current gradient value, and it decides the learning rate for each parameter. Similar to RMSProp, it uses the root mean square value of the pst gradients.

## 6. Adam:

Adam, stands for Adaptive Moment Estimation [35], is another stochastic gradient optimisation technique that uses adaptive learning rate to speed up the optimisation process. The learning rate is a hyperparameter that decides the amount of update required for each weight. The Adam uses different learning rate for each weight in the model, contrary to SGD where all weights share the same learning rate. The weights that receive very small or less frequent updates over time receives large learning rate from Adam. This speeds up the learning process as the learning rates are tuned automatically and no manual intervention is required.

In order to handle the diminishing learning rate issue, Adam uses exponential average decay for squared gradients to prevent it from accumulating over time.

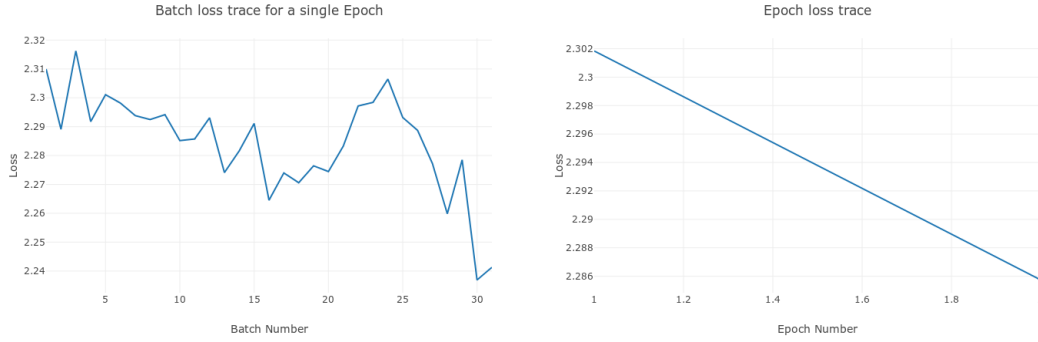
### 2.1.3 Intermediate Visualizations

The common practice during hyperparameter optimization is to evaluate the intermediate training results. The training is a costly process, and depending on the task, it could take from hours to days to complete. The analysis of the loss values during the training of the model could provide

---

<sup>7</sup>See <https://www.coursera.org/learn/neural-networks/home/welcome>

valuable insights into the final performance of the model being trained. Figure 2.4 shows the two visualizations that were made available to each participant. The first visualization plots the loss for each batch in an epoch. It is consistently overwritten with each epoch. The second visualization plots loss value for each epoch. These visualizations were updated in real-time as the model is being trained and a participant could decide to cancel the training if the participant feels the current configuration of hyperparameters is not optimal in a bid to save time and try more configurations. We used Visdom<sup>8</sup>, an open-source library from facebook, to plot the loss visualizations in real-time.



(a) The batch vs. loss graph. The x-axis depicts the batch number while the y-axis shows the loss for that specific batch. (b) The Epoch vs. Loss graph. The x-axis depicts the epoch number while the y-axis shows the loss for that specific epoch.

**Figure (2.4)** Intermediate training results.

## 2.1.4 Dataset

The task decided for this user study was object classification. The participants were asked to train a model using the given set of hyperparameters to perform object classification over ten classes. The object classification is the most prominent and classical problems that were solved using deep learning. It paved the way for the development of complex and effective deep neural networks that are being used in the majority of domains today.

The choice of the dataset was effected by the size and the difficulty of the dataset. Initially, we decided to experiment with MNIST Handwritten Digit dataset. The size of the dataset was small enough to let participants try multiple hyperparameter configurations without having to wait too long for training. Although the complexity of the dataset was lower than the required standards. The deep learning models have become so advanced that it was straightforward for the model to learn this dataset representation and perform the majority of the times accurately. In order to make the user study problem a bit more realistic, we decided to use Imagenette<sup>9</sup>.

Imagenette is a subset of the famous ImageNet dataset<sup>10</sup> [36] and is used to evaluate the credibility of the model before it is trained and tested on the ImageNet itself. It has ten classes, as

<sup>8</sup>See <https://github.com/facebookresearch/visdom>

<sup>9</sup>See <https://github.com/fastai/imagenette>

<sup>10</sup>See <http://www.image-net.org/>

shown in Figure 2.5, with each class, has 1300 training images, 50 in the validation set and test set of 50 images. Each image is of size 160 px.



(a) Class Cassette



(b) Class English springer



(c) Class Tench



(d) Class Chain saw



(e) Class Church



(f) Class French Horn



(g) Class Garbage truck



(h) Class Gas pump



(i) Class Golf ball



(j) Class Parachute

**Figure (2.5)** Imagenette Dataset example image from each class.

### 2.1.5 Model Used

The choice of the model is dependent on the type of task and dataset available. For object detection, various models have been implemented and have proven to show high performances. As each participant will train a model from scratch for every hyperparameter configuration, we wanted to find a model that is small in size, less trainable parameters but at the same time deep enough to be able to perform well for the given problem. We tried various models including *LeNet* [37], *ResNet* [38] and *DenseNet* [39] but ultimately decided to implement *SqueezeNet* [3]. The *LeNet* has a smaller size and lower accuracies while *ResNet* and *DenseNet* come with many deeper layers

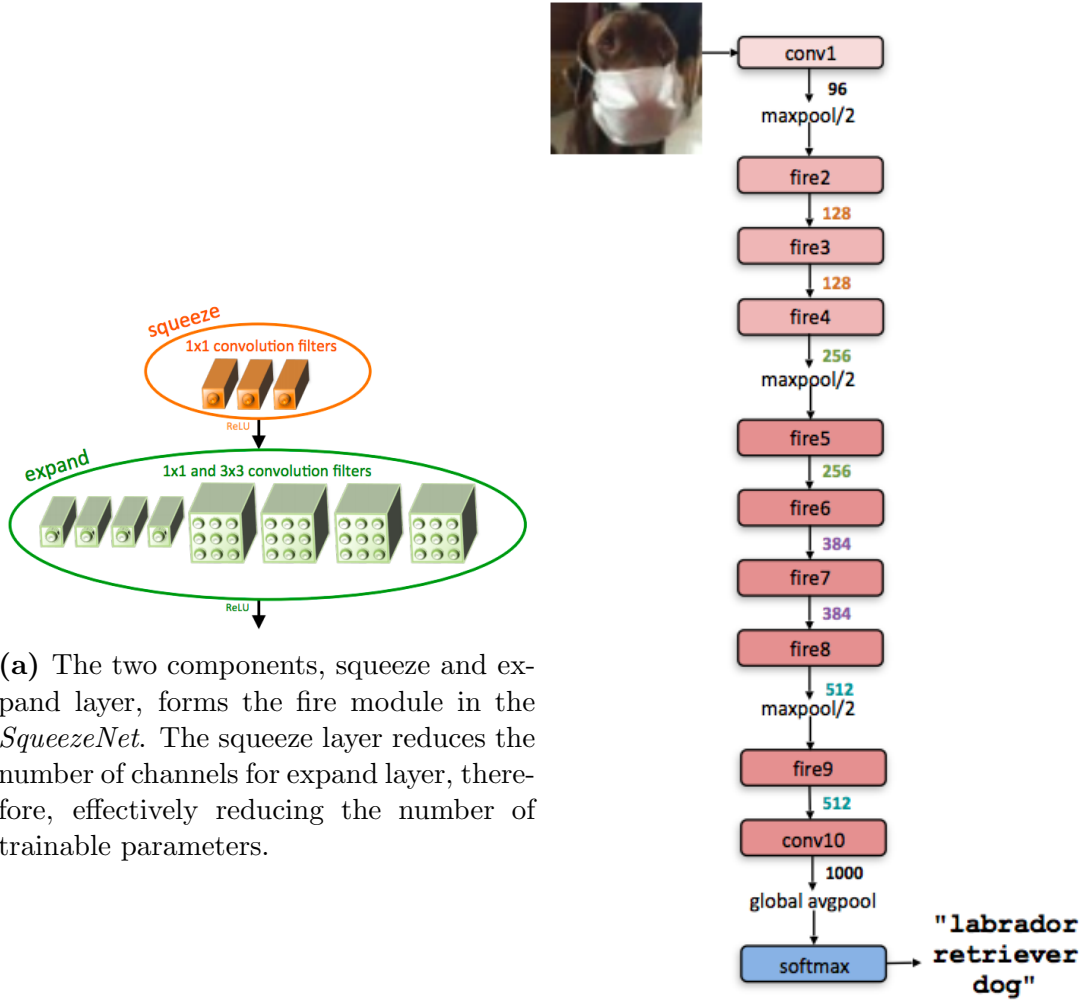
that increased the training time. The *SqueezeNet* has smaller size but similar final performances when compared to the other models.

The *SqueezeNet* reduces the size of the model by reducing 3x3 convolution filters with 1x1 convolution filters. It introduces a fire module that consists of two parts. First, the squeeze layer that is made up using 1x1 convolution filters. Next is the expand layer that is the combination of 1x1 and 3x3 convolution filters. The fire module is shown in Figure 2.7a. The squeeze layer helps in reducing the number of channel input to the expand layer reducing the number of trainable parameters in the model. Figure 2.7b shows the complete architecture of the *SqueezeNet* model. The model starts with a standalone convolution layer, stacked with eight layers of fire modules. As it can be seen from Figure 2.6, the number of filters is gradually increased in the fire modules.

layer name/type	output size	filter size / stride (if not a fire layer)	depth	$s_{1 \times 1}$ (#1x1 squeeze)	$e_{1 \times 1}$ (#1x1 expand)	$e_{3 \times 3}$ (#3x3 expand)	$s_{1 \times 1}$ sparsity	$e_{1 \times 1}$ sparsity	$e_{3 \times 3}$ sparsity	# bits	#parameter before pruning	#parameter after pruning
input image	224x224x3										-	-
conv1	111x111x96	7x7/2 (x96)	1				100% (7x7)			6bit	14,208	14,208
maxpool1	55x55x96	3x3/2	0									
fire2	55x55x128		2	16	64	64	100%	100%	33%	6bit	11,920	5,746
fire3	55x55x128		2	16	64	64	100%	100%	33%	6bit	12,432	6,258
fire4	55x55x256		2	32	128	128	100%	100%	33%	6bit	45,344	20,646
maxpool4	27x27x256	3x3/2	0									
fire5	27x27x256		2	32	128	128	100%	100%	33%	6bit	49,440	24,742
fire6	27x27x384		2	48	192	192	100%	50%	33%	6bit	104,880	44,700
fire7	27x27x384		2	48	192	192	50%	100%	33%	6bit	111,024	46,236
fire8	27x27x512		2	64	256	256	100%	50%	33%	6bit	188,992	77,581
maxpool8	13x12x512	3x3/2	0									
fire9	13x13x512		2	64	256	256	50%	100%	30%	6bit	197,184	77,581
conv10	13x13x1000	1x1/1 (x1000)	1				20% (3x3)			6bit	513,000	103,400
avgpool10	1x1x1000	13x13/1	0									
<div style="display: flex; justify-content: space-around; align-items: center;"> <span>activations</span> <span>parameters</span> <span>compression info</span> </div>											1,248,424 (total)	421,098 (total)

**Figure (2.6)** The two components, squeeze and expand layer, forms the fire module in the *SqueezeNet*. The squeeze layer reduces the number of channels for expanding layer, therefore, effectively reducing the number of trainable parameters.

The performance results comparison of *SqueezeNet* with *AlexNet* [40] on an object classification task is presented in the literature [3]. There is 50 times reduction in the size of the model in *SqueezeNet* while exceeding the Top1 and Top5 accuracy when compared to *AlexNet*. These results motivated our decision to select *SqueezeNet* as a model for our user study as it is much faster to train while providing comparable accuracy when compared to prominent models available in deep learning.



**Figure (2.7)** *SqueezeNet* model's building blocks and the architecture. The figure is reproduced from *SqueezeNet* published literature [3]

## 2.2 Design Motivation

This section lists the reason for certain choices made during the design of the user study. The user study is the main part of this research, and each component of this user study was added after careful consideration and sufficient motivation.

### 2.2.1 Web Platform

The key requirement of this user study was to replicate the hyperparameter optimization setting without having people to worry about implementing and deploying their deep learning model on the cluster. The idea was to include participants with a different background in the deep learning field, and, that includes a participant with no experience in the field of computer science. To facilitate the hyperparameter optimization task, the choice of a web platform seems appropriate. Also, to answer the question about the difference in strategies of different participants we would require detailed information about every hyperparameter configuration selected by the participant, the order in which these hyperparameter configurations were used and the participant’s reasoning for choosing those specific values. Thus, by developing a web platform, every activity of a participant can be easily recorded and managed.

This web platform is fully equipped with all the functionalities generally required during the hyperparameter optimization process. The participants could submit their choice of hyperparameters, view the intermediate visualization during training, validate their model’s performance on the validation set, and put an early stop to the training if required.

### 2.2.2 Task

The object classification is one of the most prominent and classical problems solved using deep learning. It kick started the development of deep network models like *AlexNet*, *ResNet*, *VGGNet*, et al. Thus, choosing task as object classification gave us the benefit of selecting from a variety of open-source datasets and deep learning models.

### 2.2.3 Hyperparameters

One of the drawbacks of conducting a user study to observe and analyze hyperparameter optimization is the time required for the model to complete the training. It is difficult to find participants willing to be a part of long user study. Thus, we tried including the hyperparameters that are required by the task in hand and is most commonly used. By including many hyperparameters, we could have made it inconvenient for the participants to find the optimal values for each hyperparameter in the given time. We tried replicating an object classification task and included all the hyperparameters required to train the model as efficiently as possible.

The choice of loss function and optimizers were restricted to balance the time and choices available for the participants. Also, there are a plethora of choices available for loss functions and optimizer and including them as options would require implementing each method, which makes it rather inconvenient. All the options well suited for object classification task were included.

## 2.2.4 Dataset

The selection of *Imagenette*<sup>11</sup> as our dataset for the user study was an iterative process. We first started with a simpler dataset, MNIST handwritten recognition. The training time was quite low, but the task to classify MNIST dataset was found to be quite simple. The models could easily classify images in MNIST dataset as the task was not complicated enough for the models to learn as the image size is 28px and with only two colors black and white.

The other dataset we tested was Tiny Imagenet<sup>12</sup>. This dataset has 200 classes with each class contains 600 images. Due to the size of the dataset, the training time became too high. Finally, we found a dataset that suits the object classification task, with the correct number of classes and images so that the training time is not too high and represents a real-world problem.

## 2.2.5 Deep Learning Model

The selection of SqueezeNet was influenced by two major factors: size and accuracy. The user study demanded that the model should have fewer trainable parameters but could achieve a similar level of accuracy compared to other prominent models proposed in the literature. We experimented with several models like *LeNet*, *DenseNet*, and *SqueezeNet*. As can be seen from Figure 2.8, *LeNet* could only achieve maximum accuracy of 56.08% due to the shallow architecture of the model. While *DenseNet* did improve the accuracy but at the cost of training time as it was required to use high-resolution images (320px) for *DenseNet*. The *SqueezeNet* perfectly balanced these models as it has a deeper architecture compared to *LeNet* while being easily trainable when compared to *DenseNet*. It showed high accuracy for the object classification task over *Imagenette* dataset.

## 2.2.6 Hiding Certain Information From Participants

The participants were provided with only partial information about the task and dataset. They were informed about the number of classes and size of the dataset, including the partition information of train/Val/test. As the data is open source, a simple google search could giveaway optimal hyper-parameter configurations available for this dataset. As the choice of model is majorly dependent on the type of task and dataset, information about the model was completely masked.

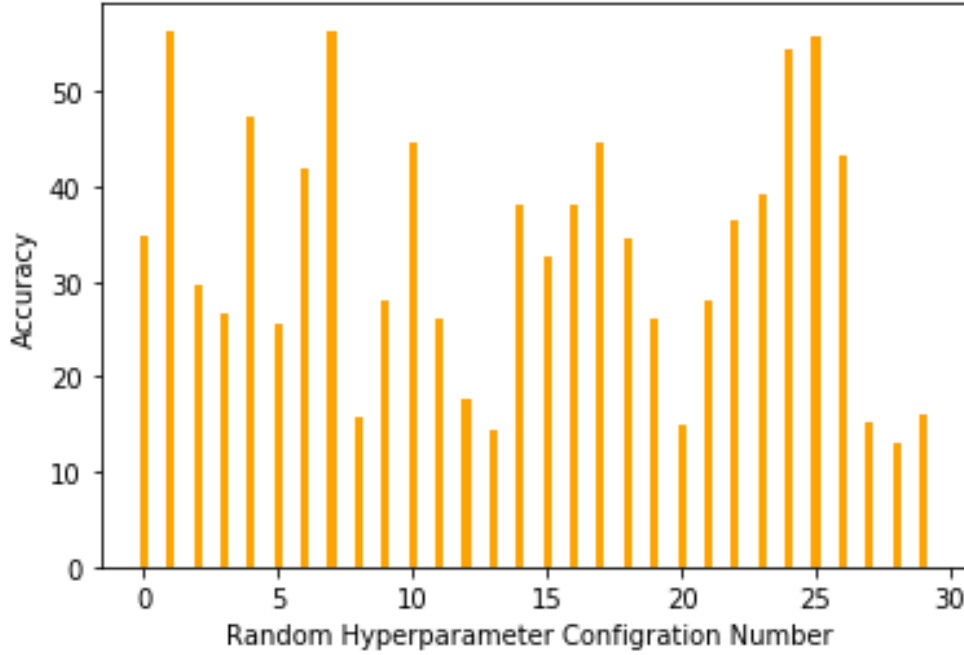
---

<sup>11</sup>See <https://github.com/fastai/imagenette>

<sup>12</sup>See <https://tiny-imagenet.herokuapp.com/>



Accuracy plot of LeNet over Imagenette for different Hyperparameter Configuration



**Figure (2.8)** The accuracy plot for *LeNet* over *Imagenette* dataset. The x-axis represent the hyperparameter configuration number found using random search and y-axis is the final accuracy for that configuration. The maximum accuracy achieved is 56.8% while the minimum being 10%

## 2.3 Participant Information

### 2.3.1 Recruiting Participants

For the user study, we conduct an open recruitment process inviting participants with different backgrounds in deep learning. We promoted this study as a contest where the participant who achieves the highest final accuracy on the test dataset wins. Apart from the first prize, there were free drinks (coffee, tea, juices) and cookies available for every participant. The monetary payment has been proven beneficial to motivate the participants and had positive effects on the participant's willingness to participate in research [41].

The recruitment process was done by advertising and promoting our user study throughout the TU Delft communications channel. The poster for our user study was displayed on all the screens of the library, 3ME, CiTG, IO, EWI, and TBM buildings. The advertisement promotes the user study and encourages students from different educational backgrounds to sign up for user study using a doodle link. The poster used for recruitment is shown in the D. During the signup process, the participants were asked to choose a 2 hour slot. It was made clear in the requirements that



a participant is free to leave at any point during the experiment, and the maximum duration was limited to 2 hours.

### 2.3.2 Participant Data

The purpose of this user study is to gather data for the analysis required to demystify the hyperparameter optimization process. There are two types of information stored for each participant - personal and experimental. The personal information includes the number of months experience in deep learning and highest education level at the time of the experiment. The experimental data contains all the hyperparameter combinations tried by the participant with the accuracy on the validation set, subjected to the model's completion of training. Every participant was asked to submit their optimal choice of hyperparameter configuration to indicate the end of the experiment. The final hyperparameter configuration for each participant is trained on the training and validation set, and the final accuracy over the test set is recorded.

Apart from this, we encouraged participants to add their motivation or line of thoughts behind choosing a certain value for a hyperparameter. The participants were provided with a predefined list of comments to choose from and an option to add their comments. The predefined comments can be viewed in Table 2.2. The predefined comments were included to make the commenting process easier for participants. Also, it is easier to analyze the predefined comments compared to personal comments, which would require much manual effort. Each comment was selected for a reason, and it is explained below:

1. **It is just a guess**

It represents that the participant selected the entered value for a specific hyperparameter as a random guess.

2. **It is a suggested default value.**

It means that the value chosen by the participant was suggested as default by the algorithm designer, or mentioned as a part of best practices online.

3. **It is the value that has worked well for me in the past**

This comment is used to show the experience of the participant. It is used when a participant uses a specific value of hyperparameter that has worked well for him in the experience, that is, for other models trained by the participant.

4. **It is the value I learned from previous submissions**

It is used when a participant finds the value for a hyperparameter that produces optimal final performance from the model. It is used to differentiate between the values learned from past experience outside the user study and the ones learned from the experiment.

5. **Other**

It allows the participant to add more detailed comments. A participant can use this option

to add any comment and explain more elaborately as to why a specific value is chosen for a hyperparameter.

At the end of the user study, 463 different hyperparameter combinations were collected from 31 participants. The final hyperparameter configuration submitted by each participant is trained and tested 10 times, and the average accuracy is reported as the final accuracy.

**Table (2.2)** Comments available for participants.

Comments
It is just a guess.
It is the suggested default value.
It is the value that has worked well for me in the past.
It is the value I learnt from previous submissions.
Other

# Chapter 3

## Experiments and Results

This chapter explains the initial experimental setup and the analysis performed over the collected data from the user study in order to answer our research questions. Firstly, we evaluate the impact of hyperparameter configurations on the final performance of the model. Secondly, we correlate the final accuracy of each participant with their months of experience in deep learning. Lastly, we explore different strategies that is used by participants and how it varies with the background in deep learning.

### 3.1 Experimental Setup

The user study was divided into multiple slots where each slot had the maximum capacity of four participants. There were four machines running independently with similar configuration as shown in the table 3.1. In order to validate the variance that could be induced due to the varying machine configurations, mainly the run time of model training, same hyperparameter configuration is used for the model and trained on four machines. The model is trained 10 times and average run time for each machine is shown in the table 3.2. The results show that there is very minimal difference between machines in terms of running time while there is no difference in the obtained final accuracy. Apart from the underlying machines, everything was same for all participants. For every statistical tests performed in following sections, we have used p-value as 0.05 to test the hypothesis, unless specified otherwise.

### 3.2 Variance Due To Hyperparameter

In this section, we first answer our first research question, that is, if there is any variance in the final performance of the model induced by varying hyperparameter configurations. The task, model and dataset is same as the one used for the user study. Our goal is to validate that by

**Table (3.1)** The hardware information of four machines used for the user study. The software configuration of all these machines are same. GB refers to Giga Byte

Machine number	processor	RAM	GPU
Machine 1	i7-3820	16GB	1GB GTX-Titan
Machine 2	i7-3820	16GB	1GB GTX-Titan
Machine 3	i7-3820	16GB	1GB GTX-Titan
Machine 4	i7-2600K	16GB	1GB GTX-Titan

**Table (3.2)** The result table showing the evaluation metric for different machines used for user study.

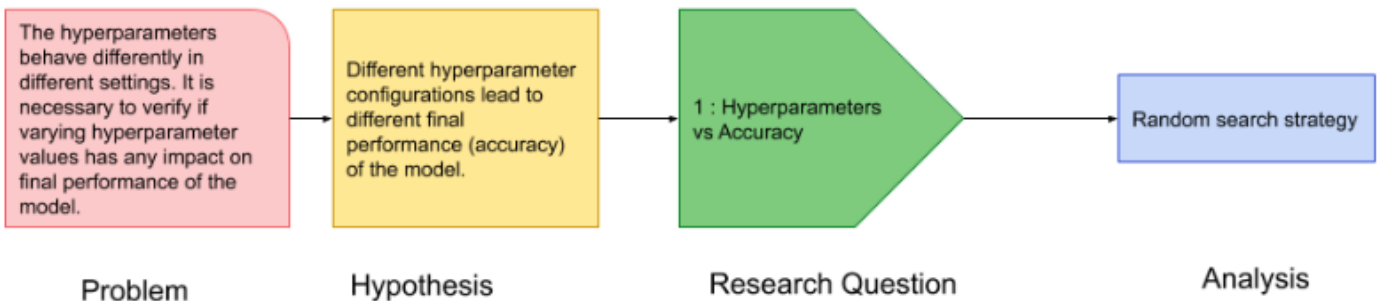
Machine number	execution time (in sec)
Machine 1	517.353
Machine 2	527.9
Machine 3	513.324
Machine 4	530.656

varying hyperparameter values, the final performance of the model changes. Thus, we used random search<sup>1</sup> to automate the process. In total of 320 hyperparameter configurations were tested and the Figure 3.2 shows the frequency histogram of the accuracy. The histogram shows the spread of final performance of the model for the given task and dataset. The final performance is highly varied, with 10–20% being the most popular range. It can be seen that maximum accuracy achieved over 320 hyperparameter configurations using random search is below 80%. The mean, standard deviation and variance of accuracy was found to be 41.83%, 25.34%, 642.52 respectively. An overview of our approach is shown in Figure 3.1

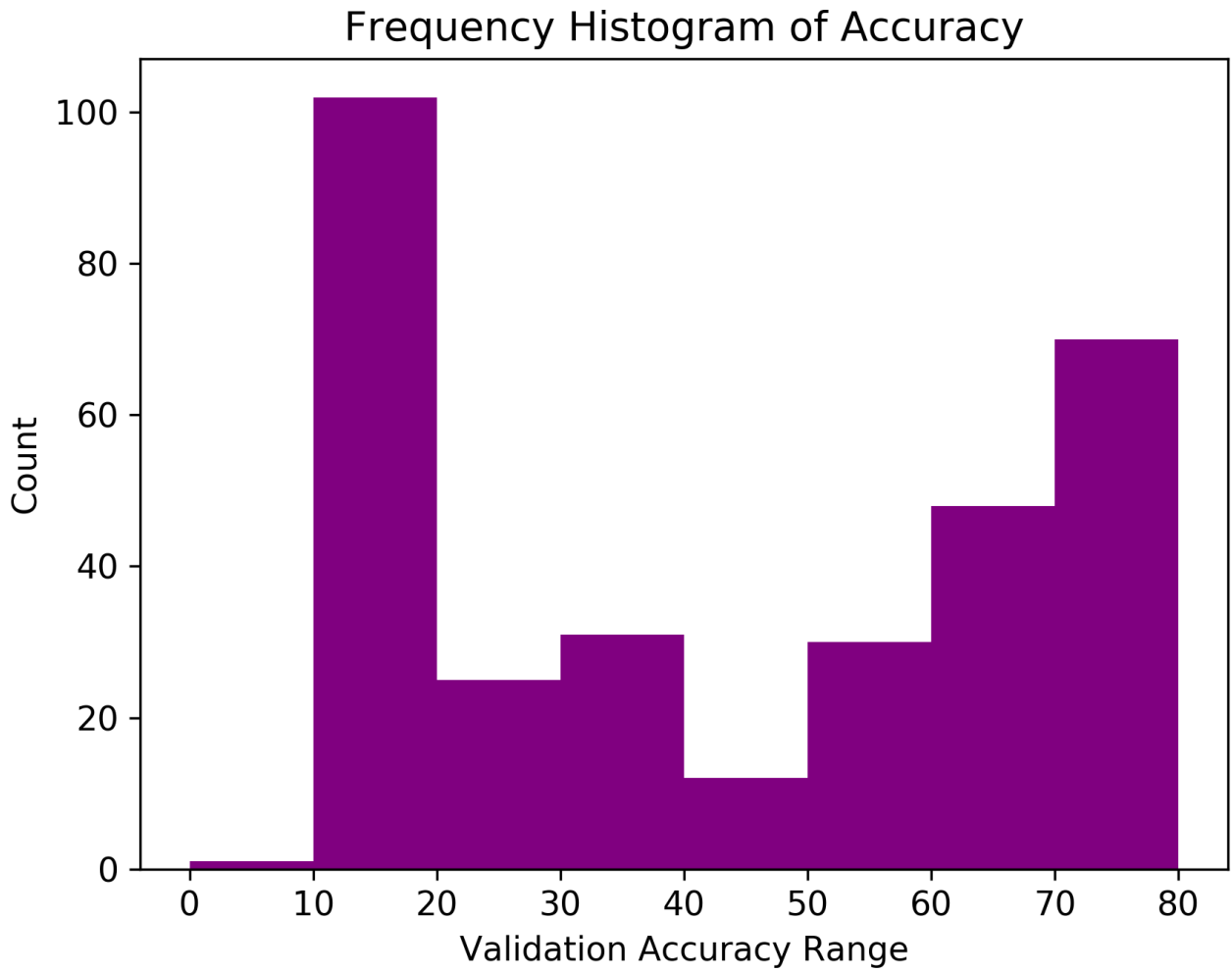
### 3.3 Final Performance and Humans

In this section, we focus on answering our second research question by using the participant’s performance data collected during user study. First, we compare the final performance achieved

<sup>1</sup>We used `sklearn.model_selection.RandomizedSearchCV` method in python.

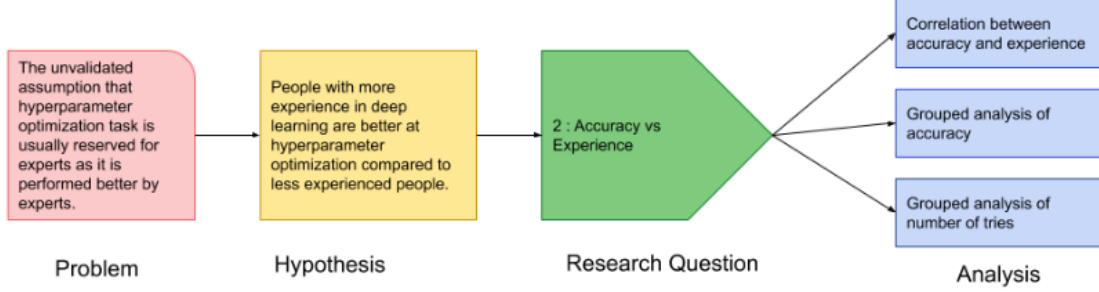


**Figure (3.1)** An overview of analysis done to answer RQ1.



**Figure (3.2)** The frequency histogram for the accuracy collected over 320 different hyperparameter configurations using random search. The x-axis depicts the accuracy bins, that is, 0-10 is a bin containing all they hyperparameter configuration resulting in final accuracy in the range  $[0,10)$ . The y-axis represents the count for that specific range. The maximum accuracy achieved is below 80% for the given model.

by participants with their experience in deep learning. Following that, we group the participants into three groups based on their experience and analyze these groups over accuracy and speed. An overview of this section is provided in Figure 3.3.



**Figure (3.3)** An overview of the analysis done for RQ2.

### 3.3.1 Accuracy vs Experience

#### Normality test

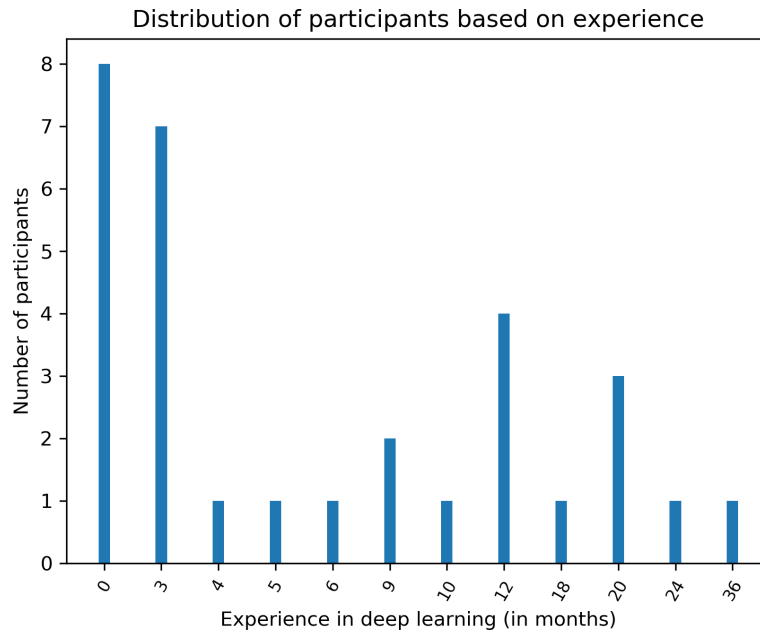
The normality test is done to validate the distribution of the data. It is important as the choice of many statistical methods are dependent on the underlying distribution of the data. We test the data collected, the final accuracy of every participant, and the experience of a participant in deep learning, for normal distribution using two methods. First, we plot a histogram and visually validate the distribution. Secondly, we use *D’Agostino and Pearson’s*<sup>2</sup> test to validate the collected data. *D’Agostino and Pearson’s* combines skew and kurtosis to create a robust test of normality [42].

Figure 3.4 shows the underlying distribution of the participant’s experience. It can be seen that the data is not normally distributed. It is supported by the statistical results obtained using *D’Agostino and Pearson’s* test as displayed in the Table 3.3. Similarly, using Figure 3.5 and Table 3.3 we can safely assume that the accuracy distribution is not normally distributed.

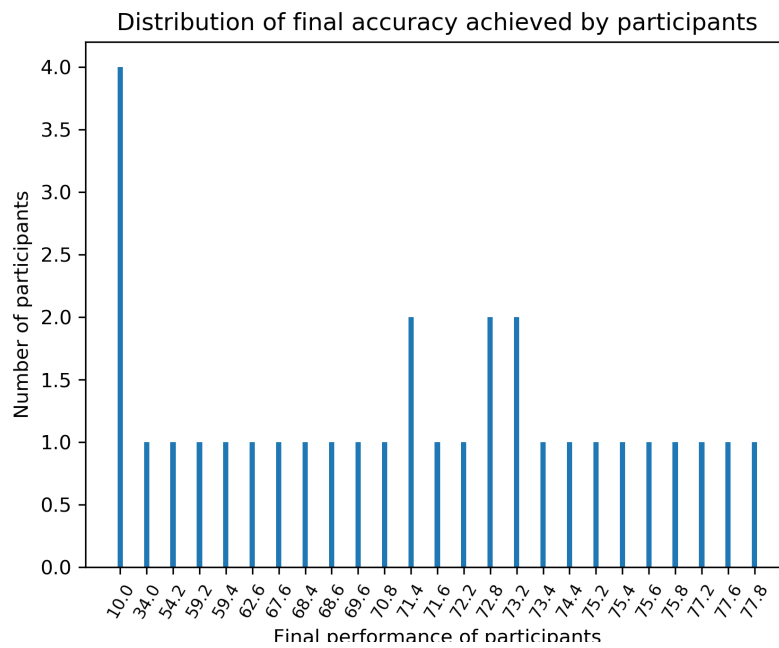
**Table (3.3)** The *D’Agostino and Pearson’s* normality test results. The null hypothesis is accepted as the p-value for both data distribution is <0.05.

Null hypothesis	test score	p-value	Decision
Participant’s experience is not normally distributed	7.56	0.02	Accept
Participant’s accuracy is not normally distributed	21.72	0.00019	Accept

<sup>2</sup>We used `scipy.stats.normaltest` method from python.



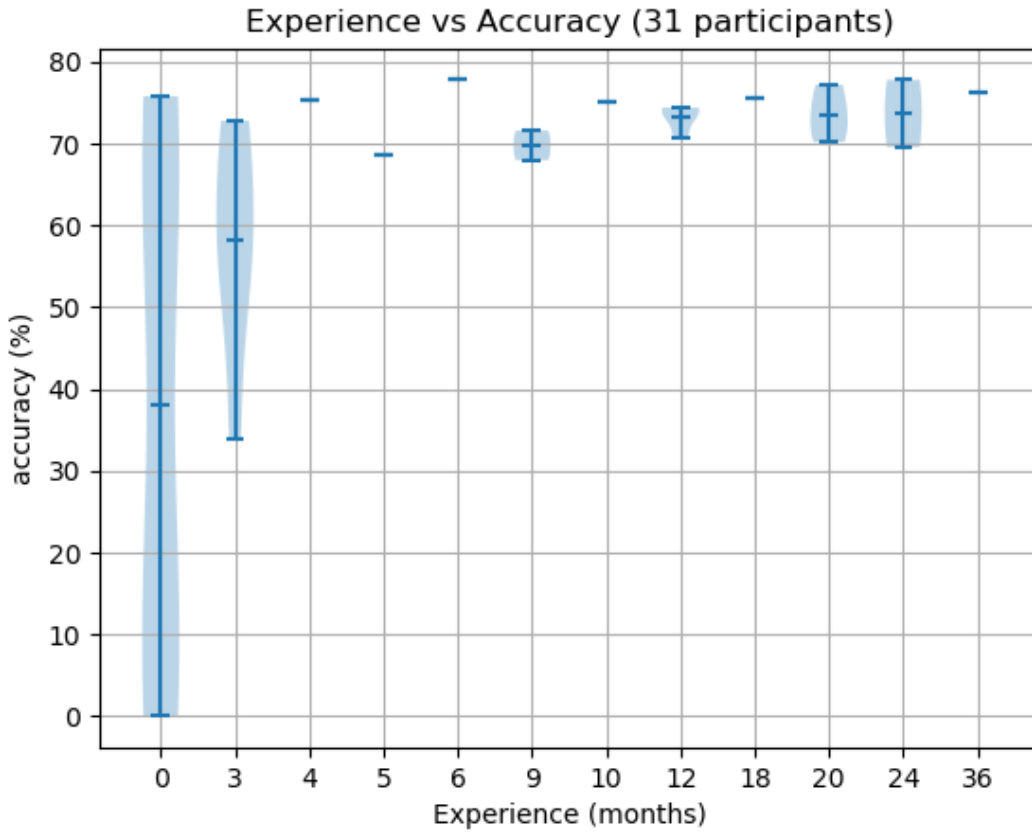
**Figure (3.4)** The distribution of experience in deep learning of participants that took part in the user study.



**Figure (3.5)** The distribution of final accuracy of participants that took part in the user study.

## Human and Hyperparameters

The final performance (accuracy) of the model was calculated on the test set using the hyperparameter configuration submitted as the final choice by each participant. In total, 31 participants completed the user study. The standard deviation, mean, and variance in the final accuracy were found to be variance is 26.318%, 55.95 and 692.65 respectively and the distribution is shown in the Figure 3.6. The The x-axis depicts the experience of the participants while the box plot shows the distribution of multiple participant with same experience and the mean. It can be seen that the final achieved accuracy tends to increase with the experience of participants. This shows that final performance of model is related to the participant training it. In order to verify the correlation between experience and final accuracy on the test set, we performed *Spearman*<sup>3</sup> correlation test. The results showed a **strong positive** correlation between experience and accuracy with a correlation factor of 0.6318 with p-value 0.00018.



**Figure (3.6)** Distribution of final performance (accuracy) of each participant.

From the Figure 3.6, we can see the jump in accuracy in participants with 4 or more number of experience in months. But the accuracy levels become similar and comparable as the experience

---

<sup>3</sup>We used `scipy.stats.spearmanr` method in python



increases. Thus, to validate if the accuracy distribution is different and vary as the experience increases, we divide the participants into three groups. The details of grouping is shown in the Table 3.4. The grouping was done based on experience and number of participants in each group.

**Table (3.4)** List of group and participants.

Group	Grouping Criteria	Number of participants
1	Experience = 0	8
2	Experience > 0 and <= 9	11
3	Experience > 9	12

### 3.3.2 Group Analysis

#### Accuracy

The first experiment was done to test the group distribution and validate the hypothesis that all three distributions are different and there is a variance that is induced due to experience. We performed levene test to validate the hypothesis as the group distributions, accuracy and experience, are not normally distributed. Levene’s test is widely used to verify if the given distributions have equal variance [43]. If the distributions are same and have equal variance, it is called as homogeneity of variance. That is, there is no variance or difference in the given distributions. The levene test can be used to verify this assumption for the distributions that is not normally distributed.

**Table (3.5)** Levene test for comparing accuracy distribution between groups divided using experience.

Groups	Score	p-value
Group 1 vs Group 2	8.40	0.01
Group 1 vs Group 3	14.338	0.001
Group 2 vs Group 3	5.52	0.029

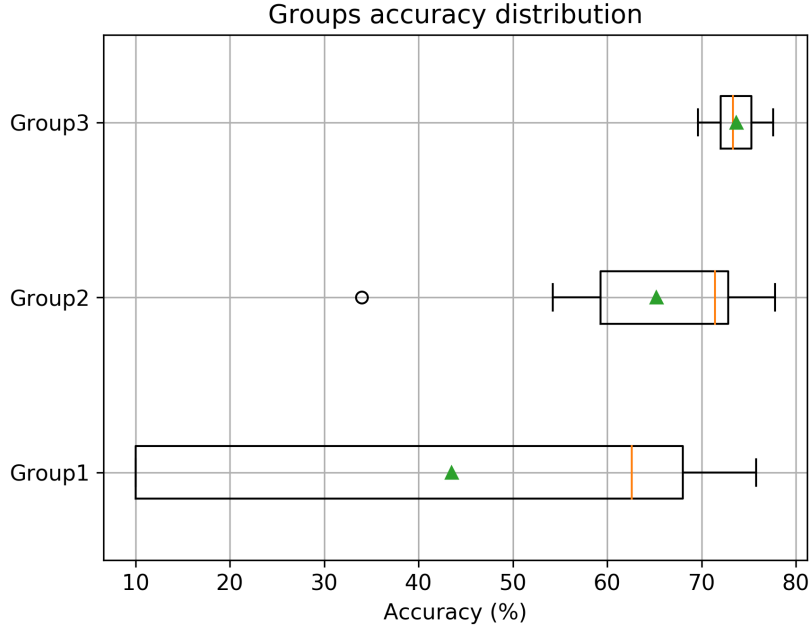
The results from Levene<sup>4</sup> test are shown in the Table 3.5. The p-value shows that it is safe to reject the null hypothesis, that is, the distributions do not have homogeneous variance. It can be seen from Figure 3.7 that Group 1 and Group3 have the highest variance and thus have the highest Levene test score.

#### Number of Turns

Now that it has been established that final performance of model is positively correlated with the experience of the participant, it was interesting to see which participants to see how many different hyperparameter configurations each participant had to try before reaching a certain final accuracy.

---

<sup>4</sup>We used `scipy.stats.levene` method in python.



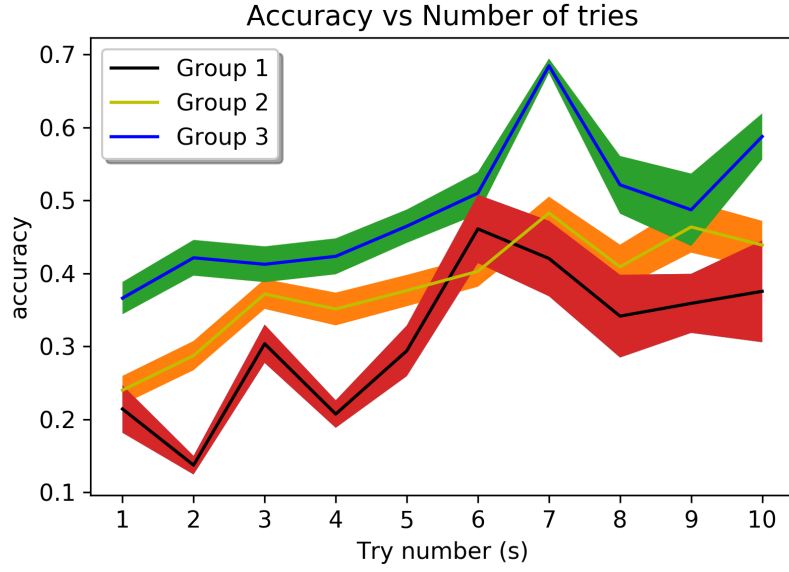
**Figure (3.7)** Distribution of final accuracy of each group. The green triangle represent the mean while yellow mark displays the median of the distribution.

It reflect the resources used by each participant to achieve the threshold accuracy. This number of tries of each participant is related to participant's experience in a bid to find any correlation. The Figure 3.9 shows the number of tries distribution for each group for three different final accuracy, that is, 25%, 50% and 75% respectively. It can be seen that Group 3 participants were quickest to improve the accuracy for all 3 different thresholds and thus using minimal resources to achieve better results. Figure 3.8 depicts the average trace of each group, that is, the average accuracy of each group after the specific number of tries. The shaded region represents the standard error of the distribution calculated using the standard deviation of the distribution.

*Thus, to answer RQ2 we can state that final performance of the model increases with the increase in participant's experience in deep learning. Also, participant with higher experience can achieve better accuracy faster compared to others.*

### 3.4 Strategies of the Experiment

In this section, we focus on answering an exploratory question. The question, What are the different strategies followed by participants and how do they evolve with experience?, aims to find insights in the strategies used by participants. This is an exploratory research question and is intended to provide better understanding of hyperparameter optimization strategies rather than conclusive evidence of which technique is better. First, we analyze the comments submitted by the participants to explain the choices made for each hyperparameter. We follow it up by the

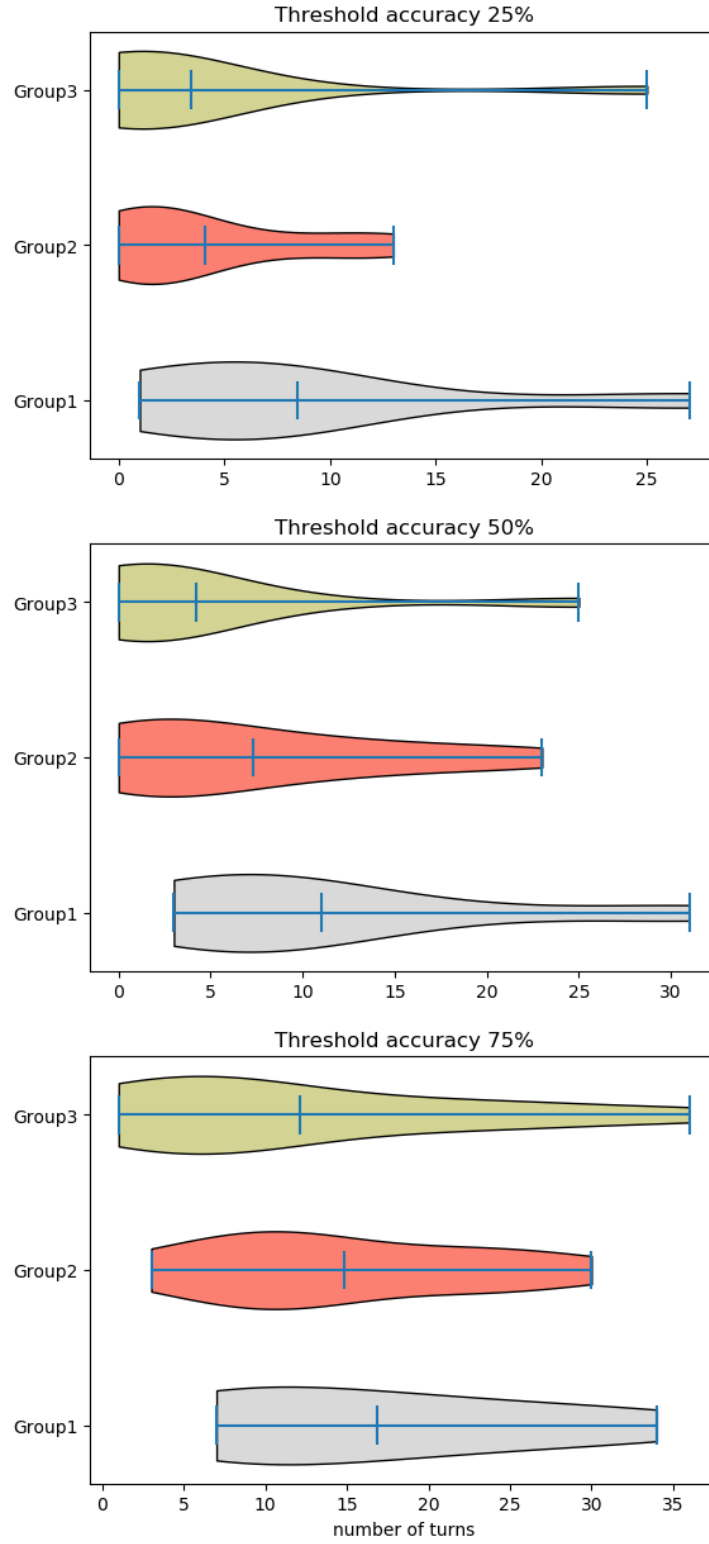


**Figure (3.8)** Average seen trace: it plots the average accuracy of the group observed after the specific number of tries. The shaded region represents the standard error for the distribution.

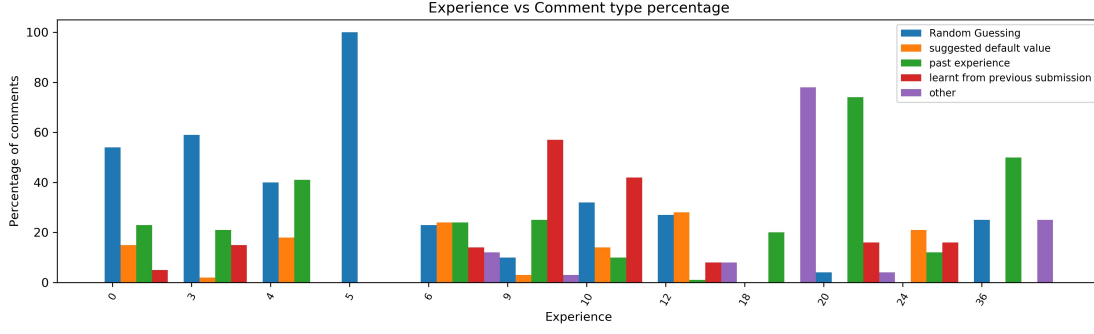
discussion on the use of default values that is suggested by the algorithm designers in the published literature.

### 3.4.1 Analysis of Comments

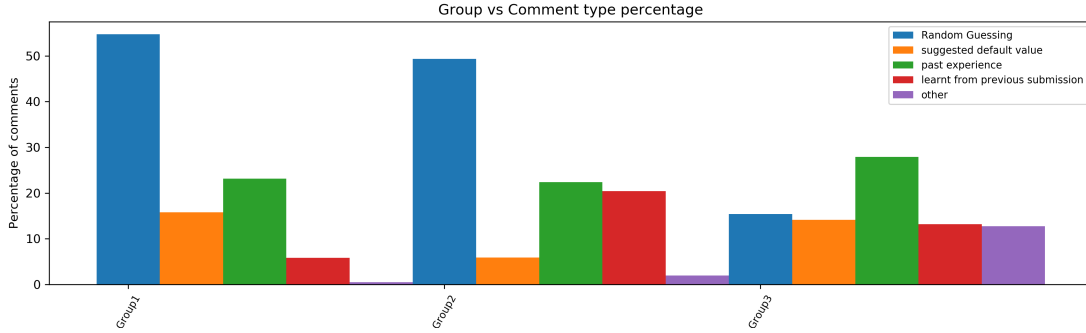
The participants were encouraged to leave comments explaining the reasoning behind choosing a specific value for a hyperparameter. In a bid to gather maximum comments, we let users choose from predefined comments explained in the Section 2.3. The Figure 3.10 shows the distribution of comments for each user and for each group, as specified in the Table 3.4.



**Figure (3.9)** Number of different hyperparameter configurations required to achieve the mentioned threshold accuracy by available groups. The mean value of each group is marked for each distribution.



(a) The batch vs loss graph. The x-axis depicts the batch number while y-axis shows the loss for that specific batch.



(b) The Epoch vs Loss graph. The x-axis depicts the epoch number while y-axis shows the loss for that specific epoch.

**Figure (3.10)** Intermediate training results.

The Figure 3.10a shows the comments percentage distribution for each experience category. The majority of comments for participant with low experience is dominated by random guessing comments which indicate the random strategy adopted by these participants. The random guessing was found to be negatively correlated with the increasing experience. We used *Spearman's* rank-order correlation and the value was found to be  $-0.58$  with p-value  $0.0007$ . The group distribution of comments show the consistent decrease in random guessing and increase in using values from past experience (green).

Submission #	Alpha.Value	Alpha.Comment.Value	Accuracy
1	0.75	2	68.6%
2	0.75	3	67.6%

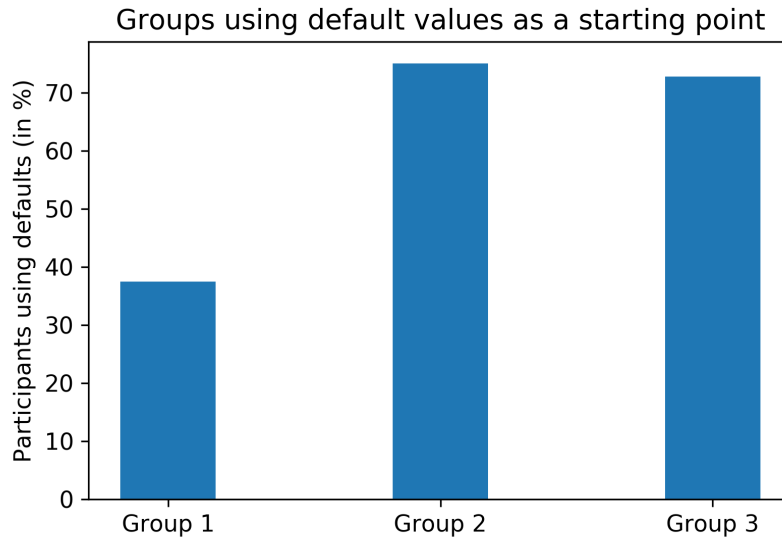
**Figure (3.11)** An example depicting how a participant with no experience in deep learning confuses between the comment type 3 and comment type 4.

From the Figure 3.10b, we can see that Group 1 which contains participants with only 0 experience

in deep learning has second highest comment percentage for past experience. This means that 22% of values used by these participants were based on their prior experience in deep learning. One explanation for this high percentage could be the confusion between past experience and learnt from previous submission. The learnt from previous submission comment is meant to indicate the values learnt from prior submissions and analyzing the results during experiments. One such example is shown in the Figure 3.11. The participant was using default value for alpha with appropriate comment (2) but for next submission updated the comment to 3 which translates to past experience instead of learnt from previous submission.

### 3.4.2 Use of suggested default values

The hyperparameters are broadly divided into two categories - optional and mandatory in our user study, as shown in Table 2.1. These optional hyperparameters are conditional; that is, they belong to a specific optimizer algorithm and come into play only when the corresponding optimizer algorithm is selected. In the user study, participants were provided the default values for these conditional hyperparameters, which were suggested by the designer of optimizer algorithms. Figure 3.12 shows the number of participants in each group using these default values as the starting point. Except for four, every participant in Group 2 and Group 3 started with all suggested default values and build on it.



**Figure (3.12)** The number of participants in each group submitting their initial hyperparameter configuration using all default values suggested by the algorithm designer.

Figure 3.12 shows the use of suggested default values by participants as the starting point. The more experienced participants tends to use these defaults more often compared to participant with no prior experience in deep learning. Although the use of defaults does not necessarily lead to

optimal hyperparameter configuration, however, all those participants who started with defaults achieved final performance greater than 50%.

# Bibliography

- [1] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.
- [2] F. Hutter, L. Kotthoff, and J. Vanschoren, eds., *Automatic Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at <http://automl.org/book>.
- [3] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size,” *arXiv e-prints*, p. arXiv:1602.07360, Feb 2016.
- [4] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, “Are GANs Created Equal? A Large-Scale Study,” *arXiv e-prints*, p. arXiv:1711.10337, Nov 2017.
- [5] R. P. A. Jasper Snoek, Hugo Larochelle, “Practical bayesian optimization of machine learning algorithms,” *Bartlett et al. [8]*, pp. 29602968, 2012.
- [6] G. Melis, C. Dyer, and P. Blunsom, “On the State of the Art of Evaluation in Neural Language Models,” *arXiv e-prints*, p. arXiv:1707.05589, Jul 2017.
- [7] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets,” *arXiv e-prints*, p. arXiv:1606.03657, Jun 2016.
- [8] D. Berthelot, T. Schumm, and L. Metz, “BEGAN: Boundary Equilibrium Generative Adversarial Networks,” *arXiv e-prints*, p. arXiv:1703.10717, Mar 2017.
- [9] D. Montgomery, “Design and analysis of experiments,” *John Wiley & Sons, Inc, eighth edn*, 2013.
- [10] R. E. B. Richard Bellman, “Adaptive control processes: A guided tour,” *Princeton University Press, 1961*, 1961.
- [11] M. Feurer and F. Hutter, “Hyperparameter optimization,” pp. 3–38.
- [12] F. Hutter, H. Hoos, and K. Leyton-Brown, “An efficient approach for assessing hyperparameter importance,” in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML’14, pp. I–754–I–762, JMLR.org, 2014.



- [13] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.
- [14] I. Loshchilov and F. Hutter, “CMA-ES for hyperparameter optimization of deep neural networks,” *CoRR*, vol. abs/1604.07269, 2016.
- [15] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, “Towards automatically-tuned neural networks,” in *Proceedings of the Workshop on Automatic Machine Learning* (F. Hutter, L. Kotthoff, and J. Vanschoren, eds.), vol. 64 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 58–65, PMLR, 24 Jun 2016.
- [16] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams, “Scalable bayesian optimization using deep neural networks,” in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 2171–2180, PMLR, 07–09 Jul 2015.
- [17] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 2951–2959, Curran Associates, Inc., 2012.
- [18] S. Falkner, A. Klein, and F. Hutter, “Bohb: Robust and efficient hyperparameter optimization at scale,” 2018.
- [19] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, pp. 5–32, Oct 2001.
- [20] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for Hyper-Parameter Optimization,” in *25th Annual Conference on Neural Information Processing Systems (NIPS 2011)* (J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberge, eds.), vol. 24 of *Advances in Neural Information Processing Systems*, (Granada, Spain), Neural Information Processing Systems Foundation, Dec. 2011.
- [21] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, “Taking the human out of the loop: A review of bayesian optimization,” *Proceedings of the IEEE*, vol. 104, pp. 148–175, Jan 2016.
- [22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.
- [23] T. Domhan, J. T. Springenberg, and F. Hutter, “Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves,” in *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI’15*, pp. 3460–3468, AAAI Press, 2015.
- [24] K. Swersky, J. Snoek, and R. P. Adams, “Freeze-thaw bayesian optimization,” 2014.

- [25] K. Jamieson and A. Talwalkar, “Non-stochastic best arm identification and hyperparameter optimization,” 2015.
- [26] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” 2016.
- [27] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” 2016.
- [28] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, “Pegasos: primal estimated sub-gradient solver for svm,” *Mathematical Programming*, vol. 127, pp. 3–30, Mar 2011.
- [29] L. Bottou and O. Bousquet, “The tradeoffs of large scale learning,” in *Advances in Neural Information Processing Systems 20 (NIPS 2007)* (J. Platt, D. Koller, Y. Singer, and S. Roweis, eds.), pp. 161–168, NIPS Foundation (<http://books.nips.cc>), 2008.
- [30] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, “Parallelized stochastic gradient descent,” in *Advances in Neural Information Processing Systems 23* (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, eds.), pp. 2595–2603, Curran Associates, Inc., 2010.
- [31] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, “On optimization methods for deep learning,” in *Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML’11, (USA)*, pp. 265–272, Omnipress, 2011.
- [32] B. T. Polyak and A. B. Juditsky, “Acceleration of stochastic approximation by averaging,” *SIAM J. Control Optim.*, vol. 30, pp. 838–855, July 1992.
- [33] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, July 2011.
- [34] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *arXiv e-prints*, p. arXiv:1212.5701, Dec 2012.
- [35] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.
- [36] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009.
- [37] Y. B. P. H. Yann LeCun, Lon Bottou, “Gradient-based learning applied to document recognition,” *arXiv e-prints*, Aug 1998.
- [38] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv e-prints*, p. arXiv:1512.03385, Dec 2015.
- [39] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” *arXiv e-prints*, p. arXiv:1608.06993, Aug 2016.

- [40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’12, (USA), pp. 1097–1105, Curran Associates Inc., 2012.
- [41] P. T. JP Bentley, “The influence of risk and monetary payment on the research participation decision making process,” *Journal of Medical Ethics*, 2002.
- [42] Z. A. Lomnicki, “Tests for departure from normality in the case of linear stochastic processes,” *Metrika*, vol. 4, pp. 37–62, Dec 1961.
- [43] I. Olkin, “Contributions to probability and statistics; essays in honor of Harold Hotelling,” *Stanford, Calif., Stanford University Press, 1960.*, 1960.
- [44] *Spearman Rank Correlation Coefficient*, pp. 502–505. New York, NY: Springer New York, 2008.

# Appendix A

## Statistical methods used for analysis

### A.1 Spearman rank-order correlation method

The *Spearman* [44] correlation is a non-parametric test for evaluating the strength of a monotonic relationship between two variables. The non-parametric test does not make any assumption about the underlying distribution of the variables. Spearman's correlation coefficient, represented as  $r_s$ , is the strength of the monotonic relationship between two variables and ranges between  $[-1,1]$ . The interpretation of different values of  $r_s$  is shown in the Table A.1. The strength of the relationship ( $r_s$ ) is calculated as [44]:

$$r_s = 1 - \frac{6 \sum d_i^2}{n^2(n-1)} \quad (\text{A.1})$$

where  $d_i$  is the difference between the ranks of two variables and  $n$  is the total number of data points.

**Table (A.1)** Interpreting different values of  $r_s$ .

$r_s$ value	Interpretation
$r_s = 0$	No monotonic relationship
$r_s < 0$	Negative monotonic relationship between two variables
$r_s > 0$	Positive monotonic relationship

### A.2 Levene's test

*Levene's* test [43] is performed to validate if different groups containing multiple samples have homogeneity of variance. It is a non-parametric test that could be used to analyze the variance

between groups and find the difference in variance. The *Levene's* test assumes the null hypothesis that all groups have similar variances while the alternative hypothesis is that there is at least a pair of groups that has different variance.

# Appendix B

## Final submission of each participant

**Table (B.1)** The final submission of each participant. List of abbreviations used for the headings are listed in Table B.2

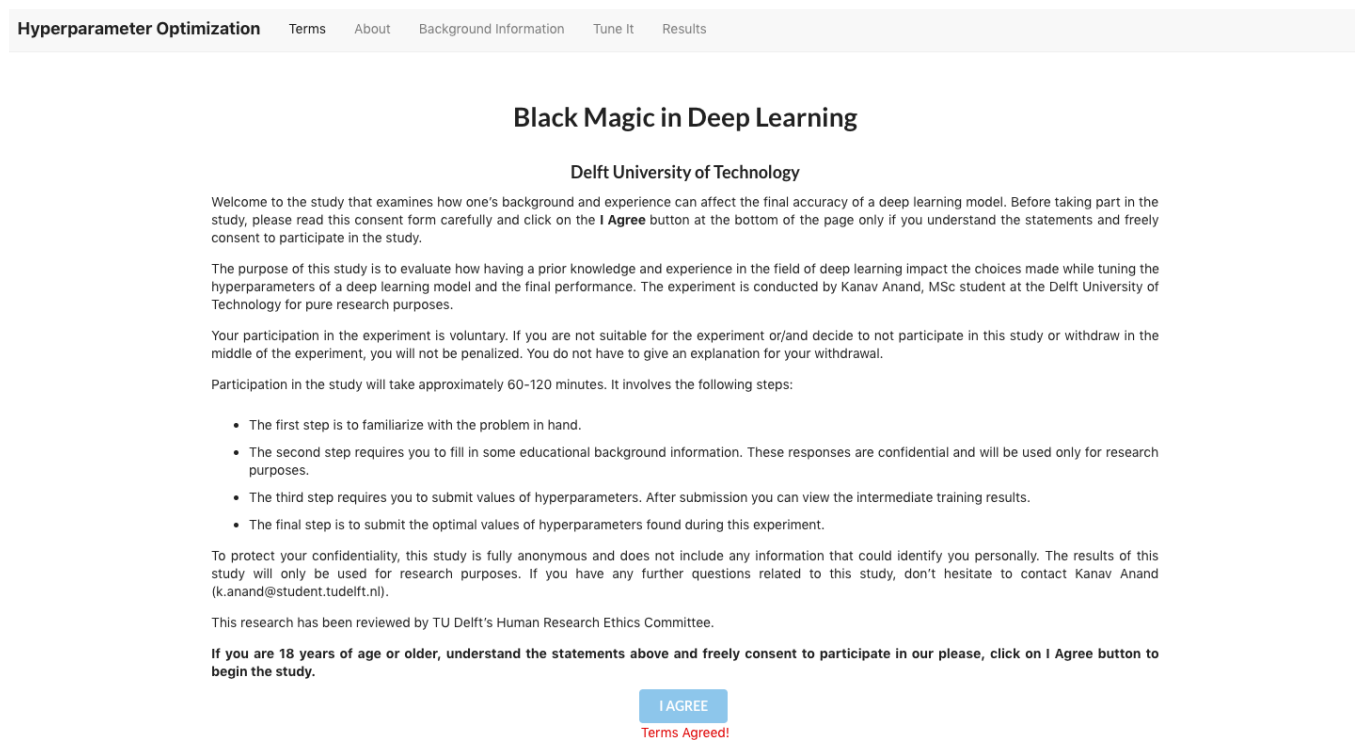
	Alpha	BS	B1	B2	EPOCH	EPS	IA	LAMBD	LR	LRD	LF	MOM	OPT	RHO	UserID	WD
1	0.99	30	0.9	0.999	100	1e-06	0	0.0001	0.0001	0	cross_entropy	0	adam_optimizer	0.9	BC7935A	0
2	1	10	0	0	12	1e-10	0	0.0001	0.01	0	mean_squared_loss	0	adam_optimizer	0.9	BB8DA7A	0.9
3	0.99	10	0.9	0.999	20	0.1	0	0.0001	0.001	0	cross_entropy	0	adam_optimizer	0.9	7D4253A	0
4	0.99	64	0.9	0.999	50	0.01	0	0.0001	0.001	0	cross_entropy	0	adam_optimizer	0.9	A86DC8A	0.001
5	0.99	25	0.9	0.999	150	1e-06	0	0.0001	0.001	0	cross_entropy	0	averaged_sgd	0.9	6B392DA	0
6	0.99	32	0.9	0.999	25	1e-06	0	0.0001	0.0001	0.01	cross_entropy	0	adam_optimizer	0.9	5B55DDA	0.001
7	0.99	100	0.9	0.999	75	1e-06	0	0.0001	0.001	0	cross_entropy	0.95	sgd	0.9	303272A	0
8	0.99	300	0.9	0.999	200	1e-06	0	0.0001	0.001	0	cross_entropy	0	averaged_sgd	0.9	347864A	0
9	0.99	100	0.9	0.999	25	1e-06	0	0.0001	0.001	0	mean_squared_loss	0	adam_optimizer	0.9	02AECEA	0
10	0.99	32	0.9	0.999	32	1e-06	0	0.0001	0.0005	0	cross_entropy	0	adam_optimizer	0.9	BEBF04A	0.1
11	0.99	200	0.9	0.999	10	1e-06	0	0.0001	0.001	0	cross_entropy	0	adam_optimizer	0.9	9950E8A	0
12	0.99	100	0.9	0.999	100	1e-06	0	0.0001	0.001	0	mean_squared_loss	0	rms_prop	0.9	B65339A	0
13	0.99	10	0.9	0.999	100	1e-06	0	0.0001	0.01	0	cross_entropy	0	averaged_sgd	0.9	DC871EA	0
14	0.99	130	0.9	0.999	150	1e-06	0	0.1	0.0001	0	cross_entropy	0.5	adam_optimizer	0.9	408EECA	1e-05
15	0.99	13	0.9	0.999	18	5e-06	0	0.0001	0.05	0	cross_entropy	0	ada_delta	0.9	193DD8A	0.0001
16	0.99	128	0.9	0.999	80	1e-06	0	0.0001	0.001	0	cross_entropy	0.9	sgd	0.9	45A082A	0.0005
17	0.99	1	0.9	0.999	6	1e-06	0	0.0001	0.001	0	mean_squared_loss	0	ada_grad	0.9	6EBC93A	0
18	0.99	32	0.9	0.999	60	1e-06	0	0.0001	0.001	0	cross_entropy	0	sgd	0.9	114076A	5e-05
19	0.99	200	0.9	0.999	1000	1e-06	0	0.0001	0.5	0	cross_entropy	0.2	sgd	0.9	44B94AA	0.25
20	0.99	200	0.99	0.99	200	0.0001	0	0.99	0.00015	0.6	cross_entropy	1.2	sgd	0.9	CB4504A	0.9
21	0.99	480	0.9	0.999	100	1e-06	0	0.0001	0.001	0	negative_log_likelihood	0	adam_optimizer	0.9	751A69A	0
22	2	800	0.8	0.8	1200	6	0	0.0001	8	0	cross_entropy	4	averaged_sgd	0.9	DF66BFA	8
23	0.99	512	0.7	0.999	700	1e-06	0.1	0.0001	0.001	0.001	cross_entropy	0.9	sgd	0.8	582898A	0
24	0.99	10	0.9	0.999	50	1e-06	3	0.0001	0.01	10	mean_squared_loss	0.9	sgd	0.9	088842A	0.001
25	0.99	50	0.5	0.999	50	1e-06	0	0.0001	0.0002	0	cross_entropy	0.9	adam_optimizer	0.9	C52FFBA	0
26	0.99	50	0.9	0.999	100	1e-06	0	0.0001	0.001	0	cross_entropy	0	sgd	0.9	739CF5A	0
27	0.99	128	0.9	0.999	50	1e-08	0	0.0001	0.0001	0	cross_entropy	0	adam_optimizer	0.9	F16B6BA	0
28	0.99	950	0.9	0.999	1000	1e-06	0	0.0001	0.36	0	cross_entropy	0.9	adam_optimizer	0.9	DAC1D1A	0.7
29	0.99	555	0.9	0.999	70	1e-06	0	0.0001	0.001	0	cross_entropy	0	adam_optimizer	0.9	51DAACA	0
30	0.99	900	0.9	0.99	600	0.1	0	0.0001	0.02	0	cross_entropy	0	ada_delta	0.95	342496A	0.01
31	0.75	32	0.9	0.999	70	1e-06	0	0.0001	0.003	0	cross_entropy	0.1	sgd	0.9	0B7093A	3e-05

**Table (B.2)** List of abbreviations used to display the final submission table.

Abbreviation	Complete Name
Alpha	Alpha
BS	Batch Size
B1	Beta 1
B2	Beta 2
EPS	Epsilon
IA	Initial accumulator
LAMBD	Lambda
LR	Learning rate
LRD	Learning rate decay
LF	Loss function
MOM	Momentum
OPT	Optimizer
RHO	RHO
UserID	UserID
WD	Weight decay

# Appendix C

## User study screenshots



**Figure (C.1)** The terms and condition webpage used for the user study. It clearly states all expectations and requirements for participants.



**Background Information**

Experience with Deep Learning (in months)

2

Highest education level:

Bachelors

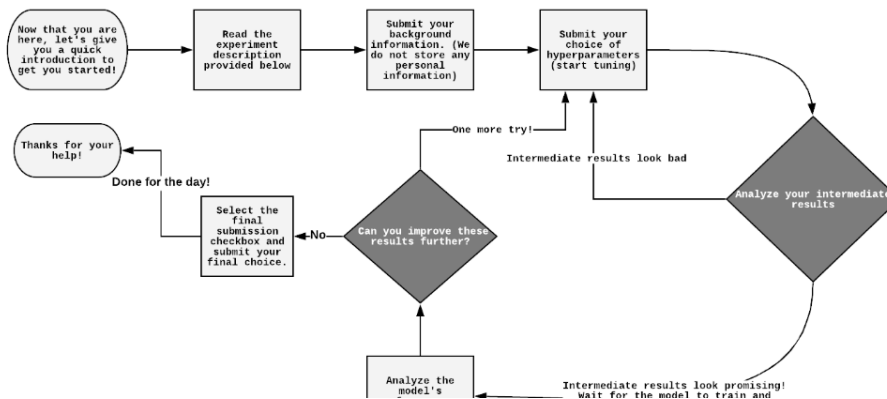
**Figure (C.2)** The background information page is shown above. It asks the participants to enter their relevant experience in the field of deep learning and their highest completed education level. Once the information is submitted by the user, anonymous user id is generated to identify every unique participant.

## Experiment Details

### Task in hand

The task is to find the optimal set of hyperparameters maximizing the final performance metric, that is, the accuracy of the model on the test set. You will be asked to submit the values of different hyperparameters exposed by the model using a form. We encourage you to use the comments field next to each hyperparameter to submit your line of thoughts. Upon submission you can view intermediate performance (loss value) of the model on the training data for each batch and epoch. You can use these intermediate results to pre-maturely end the training and submit new choice for the hyperparameters. After training of the model is finished, the accuracy of the model on validation set is displayed in the results tab.

This task can be repeated over multiple times until you think the performance cannot be improved further by updating Hyperparameter values or time limit is reached. In order to submit your final choice of hyperparameters, select the checkbox next to the submit button.



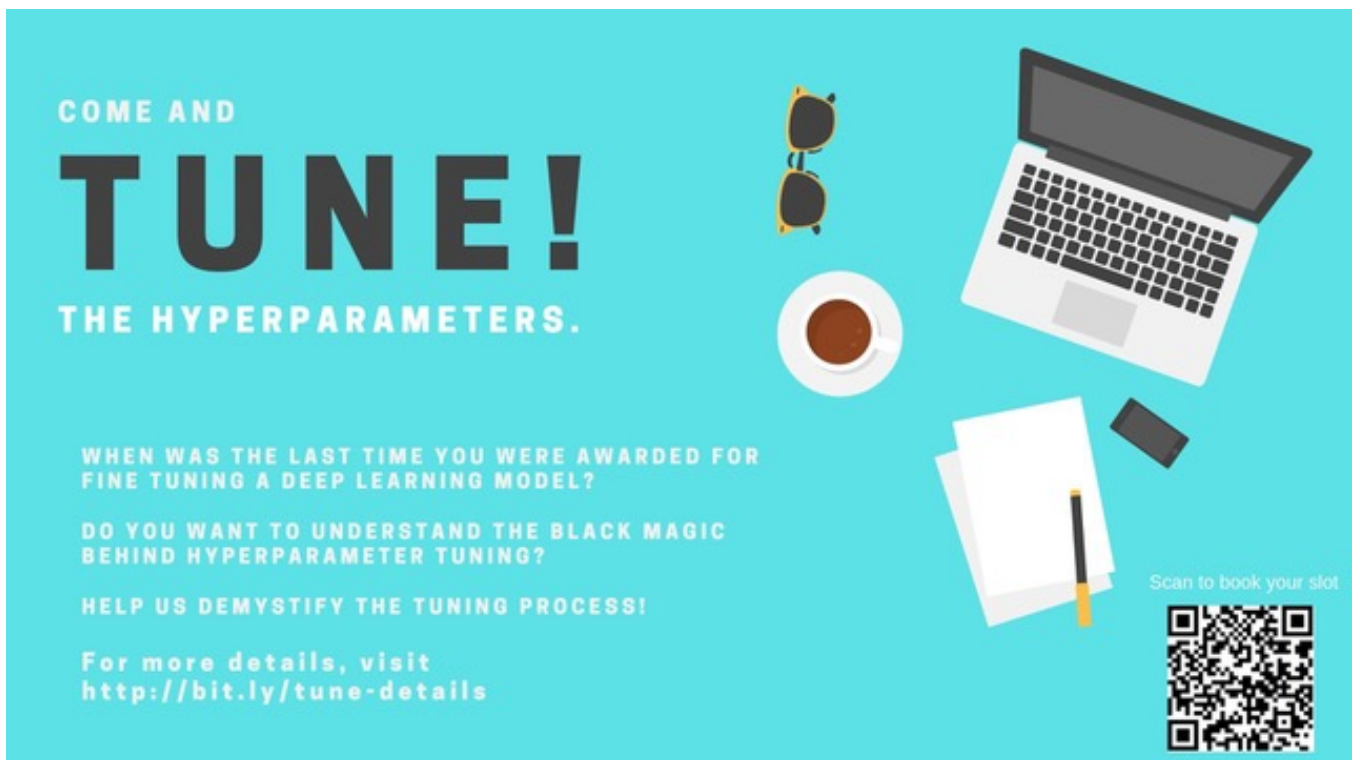
**Figure (C.3)** The experiment details page provides the detailed information of the user study. It starts with the flow diagram of the user study and follows it up with the explanation of every hyperparameter used for the experiment.

Epochs:	<input type="text" value="1"/>	<input type="text" value="Select your Comment"/>
Batch Size:	<input type="text" value="1"/>	<input type="text" value="Select your Comment"/>
Loss Function:	<input type="text" value="Cross Entropy"/>	<input type="text" value="Select your Comment"/>
Optimizer:	<input type="text" value="Adam Optimizer"/>	<input type="text" value="Select your Comment"/>
Learning Rate:	<input type="text" value="Enter Learning Rate"/>	<input type="text" value="Select your Comment"/>
Epsilon:	<input type="text" value="Enter Epsilon Value"/>	<input type="text" value="Select your Comment"/>
Weight Decay:	<input type="text" value="Enter Weight Decay Value"/>	<input type="text" value="Select your Comment"/>
Beta1:	<input type="text" value="Enter Beta1 Value"/>	<input type="text" value="Select your Comment"/>
Beta2:	<input type="text" value="Enter Beta2 Value"/>	<input type="text" value="Select your Comment"/>
<input type="button" value="Submit Job"/>		<input type="checkbox"/> Select this to indicate your final submission

**Figure (C.4)** The hyperparameter submission form is used by the participant to submit the values for hyperparameters and their corresponding comments. Once a hyperparameter configuration is submitted, the user can view intermediate results and early stop the training of the model.

# Appendix D

## Poster and Flyer



**Figure (D.1)** Poster used to display on the screens of majority of faculty buildings at TU Delft.



**Figure (D.2)** Flyer used to recruit people for user study.