# Can object-specific frequency information from labeled data improve a CNN's robustness to adversarial attacks?

Nikolaos Mertzanis, Nergis Tömen, Yancong Lin, Attila Lengyel

**Abstract**

Convolutional Neural Networks are particularly vulnerable to attacks that manipulate their data, which are usually called adversarial attacks. In this paper, a method of filtering images using the Fast Fourier Transform is explored, along with its potential to be used as a defense mechanism to such attacks. The main contribution that differs from other methods that use the Fourier Transform as a filtering element in neural networks is the use of labeled data to determine how to filter the pictures. This paper concludes that, while the filtering proposed is hardly better than a simple low-pass filter, it still manages to improve resistance to adversarial attacks with a minimal drop in the standard accuracy of the network.

## 1 Introduction

### 1.1 Problem Description

Adversarial attacks refer to the manipulation of data fed to a neural network to cause the network to classify it incorrectly. This manipulation is usually done by adversarial perturbations, which refer to high-frequency transformations applied to data that are later classified by the network. These "attacks" usually construct an image almost identical to its unfiltered counterpart when observed by a human eye but are exceptionally good at causing neural networks to falsely classify the content contained in the image. Adversarial attacks are a problem with almost any type of neural network since the attacker can very easily fool the network and even cause it to classify data as a specific class of their choice with high confidence. One of the reasons they are worth investigating is that while they are easy to generate, they are difficult to defend against. Finally, adversarial attacks are intriguing because they "expose a fundamental blind spot in our training algorithms" [7], one that can be used to target class-specific features learned by a network, invisible to humans [10]. Below is an example of such an attack.

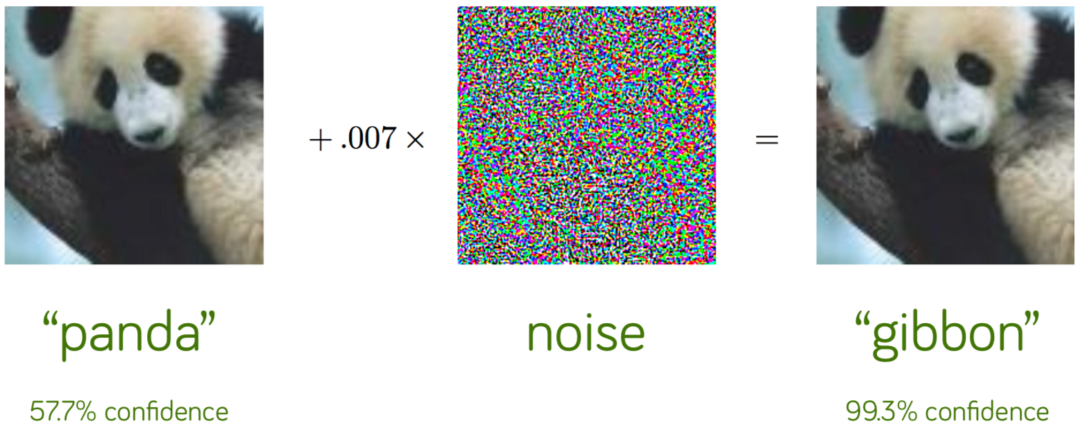


Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet [7]

## 1.2 Approach & Findings

This paper is examining how one can improve a convolutional neural network's robustness, meaning resistance to adversarial attacks, using transformations dependent on frequency information. The adversarial attacks are mocked by introducing high-frequency noise in the images. Then, an attempt is made to construct a frequency "mask" which is used to remove the unnecessary frequencies generated by these adversarial examples. This mask is applied to the data of one version of the network, both during training and test time, while no mask is be applied in another. This other, plain network, is also fitted with a simple low-pass filter to construct a third network with a very simple adversarial defense. The accuracy of classification is then measured and compared between the three (a graphical demonstration of this approach is included in Figure 5). A drop in performance is observed on both networks utilizing adversarial defenses. Then, black-box attacks are mocked on the dataset and the performance is once again compared between the networks. This time a significant performance improvement is observed on the networks utilizing the adversarial defenses, although the proposed "smart" mask constructed is not significantly better than the simple filter.

Previous work in this area has indicated multiple methods that have successfully been able to improve robustness such as adversarial training, model ensemble and high-frequency suppression [19]. This paper will be utilizing a form of the last one, which is high-frequency suppression. Ideally, a network made as "robust" as possible would be constructed using a combination of the three aforementioned methods since they all indicate an improvement in performance for adversarial examples [19]. At the same time, in order to optimize this resulting adversarial performance each of the components that are used as "defenses" need to be optimized, in ways such as the one proposed in this paper.

The network used in the paper is an existing implementation utilizing a popular convolutional network called YOLO [15]. The dataset this network is performing object detection on is taken directly from a Kaggle challenge called "Global Wheat Detection" [6]. In this challenge, participants are required to construct an object detector that can detect wheat heads from images. Currently, the best performing network in the Kaggle challenge has 77% accuracy and is based on the YOLOv5 network which is the network that is used for this paper.

Lastly, there are further sub-topics related to the research question. These relate to the method for determining a high-performing frequency mask construction, which is described in section 3, as well as the trade-off between robustness and accuracy. The latter refers to the fact that even when utilizing the most sophisticated of adversarial defenses, one is still expected to observe a drop in the non-adversarial/standard accuracy of the network [18]. This trade-off is not a necessary condition and, ideally, through rigorous and careful mask construction accuracy improvements could also occur since the filtered images may contain less unrelated data that the network has to process. This was not proven to be true in the case of this paper but, to the contrary, a drop in accuracy was observed. At the same time, there is a clear improvement in the robustness of the network using the mask proposed in this paper, although it is almost negligibly small when compared to a simple low-pass filter.

## 2 Background and related work

### 2.1 Review of adversarial attacks

Before delving into the topic of this paper, an extensive amount of research was required to first understand the nature of adversarial attacks and then construct the pipeline described in sub-section 3.2. Adversarial examples have been discovered since at least 2014 [7] but have mostly been a topic of study in recent years.

There are two types of adversarial attacks depending on the adversarial knowledge available to the attacker:

- **White-box**: adversarial attacks where the attacker has access to the model's weights and parameters and can hence optimize the noise construction to target properties specific to the network [14]. These include attacks that require multiple iterations to be constructed, such as Projected Gradient Descent [12] and the Fast Gradient Sign Method [3]. White-box attacks are the most sophisticated method of the two but are also hard to construct since they require iterative processes which learn based on the output of the network. This is why, for the purposes of this paper, no white-box attacks are performed.
- **Black-box**: attacks for which transformations applied on the images do not require specific knowledge about the network that will interpret them. These can be white noise or simple rotations and translations of the image data. They generally perform worse than white-box attacks but are still capable of disrupting the performance of a network significantly [5]. In this paper, two black-box attacks are performed. One uses a simple white-noise package called "Noisify" [4] to introduce noise in the images while the other uses the SciPy package [16] to introduce Gaussian noise in the images by randomizing the pixel values (as described in section 5).

There are further divisions of attacks which can either belong to the black-box or the white-box category and depend on the goal of the attack [14]:

- **Targeted**: these attacks aim to cause the network to misclassify the input to a specific target class. As mentioned above they can be either black-box or white-box but it is natural to assume that knowledge regarding the network's weights and structure can allow for much more precision when it comes to targeted attacks.

- **Untargeted**: this type of attack simply aims to cause the network to misclassify the input.

Finally, depending on the perturbation scope of attack, it can be considered a universal or an individual attack [14]:

- **Universal**: the adversarial perturbation performed is common for the whole dataset.
- **Individual**: the adversarial perturbation performed is tailored to a specific sample from the dataset.

For the example setting explored in this paper, since there is only one class to be detected by the networks, any attack performed can be considered untargeted. Also, as was stated earlier, the attacks performed will be black-box attacks and even though the noise introduced in every image is different, the process of generating the noise is the same for every sample. Hence, they are also universal attacks.

## 2.2 Frequency Analysis

Similarly, as there multiple types of attacks, there are multiple ways of defending a network against adversarial attacks. Such are adversarial training, high-frequency suppression, model ensemble [19] and others. All of them are capable of increasing robustness, which is resistance to adversarial attacks but for the purposes of this paper, only high-frequency suppression will be used.

The high-frequency suppression is encapsulated in the use of the Fast-Fourier Transform (FFT). It is a very efficient algorithm that can compute the Discrete Fourier Transform of a sequence and its inverse. In layman's terms, it can decompose data into the frequencies it is made up of and, vice versa, it can reconstruct the original data from those frequencies. Based on that decomposition, specific frequencies will be selected to be removed from the images before reconstructing the remaining frequencies. Similar methods have already been implemented to remove noise from seismic signals [8] as well as "to learn features directly in the frequency domain" [17].

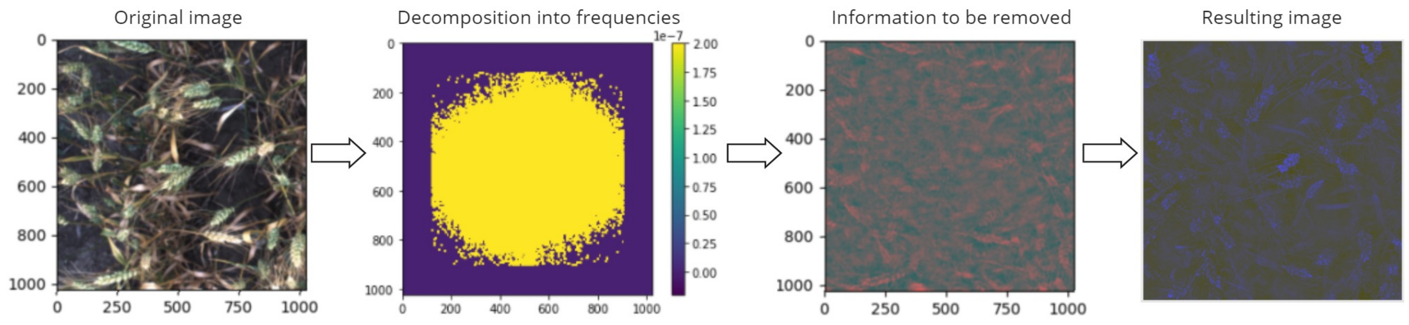


Figure 2: Example of frequency transformation pipeline. Depending on the labeled bounding boxes of the wheat heads, thresholds are set in the FFT decomposition which lead to removal of theoretically unnecessary frequencies. For example, most of the red colour in the image.

## 2.3 Network Architecture

The last part of the related work that should be mentioned is the convolutional neural network that will be used, which is YOLOv5 ("You Only Look Once: Unified, Real-Time Object Detection" [15]). This network was first created in 2015 by Joseph Redmon, et al. and outperformed all other object detection methods at the time ([15], abstract), while at the same time being relatively fast and even able to process video at 45 frames per second [15]. It is also an easy to set up network and performed exceptionally well in the Kaggle challenge the dataset was taken from [6] which made it the front-runner choice.

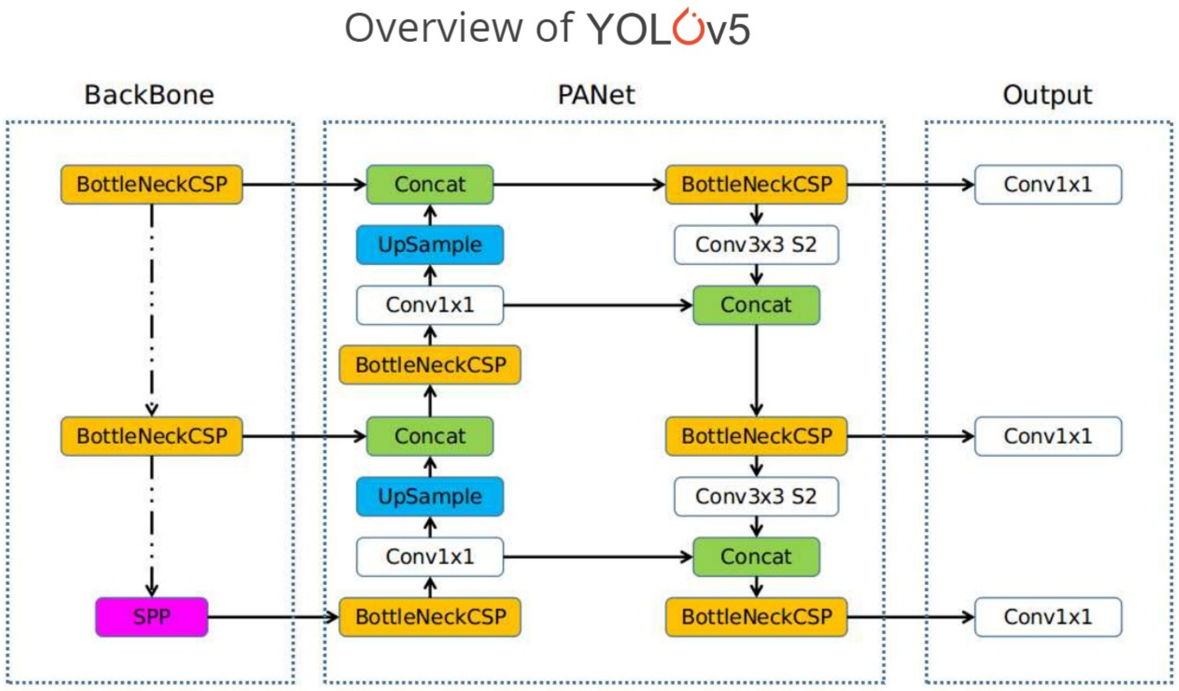


Figure 3: Overview of the YoloV5 network architecture. [15], [13]

As is visible from the architecture overview, the network can be divided into 3 main parts [2].

- **Backbone**: A convolutional neural network that applies different transformations on the images and forms features based on the data.
- **Neck (PANet)**: Layers that mix and combine features so that they can be passed forward and classified.
- **Head/Output**: Based on features from the neck it determines box (bounding box location) and class prediction steps. (Class prediction is meaningless for our purpose since there is only one class, namely, wheat heads).

The official model summary [15] states that the small YoloV5 model has 191 layers, 7.46816e+06 parameters and 7.46816e+06 gradients. Lastly, the loss function of YOLOv5 is a multi-component loss function which is the same as in YOLOv3.

Regression loss

$$\boxed{\begin{aligned}&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\right] \\ &+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\right]\end{aligned}}$$

Confidence loss

$$\boxed{\begin{aligned}&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\ &+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2\end{aligned}}$$

Classification loss

$$\boxed{+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2}$$

Figure 4: Loss function of YOLOv5. [1]

Here, regression loss refers to the difference of the predicted bounding box with the labeled data. Confidence loss relates to the confidence with which an object is assigned to a specific class, while classification loss relates to whether the class itself is the correct one. The last part of the equation is irrelevant for this case since there is only one class, namely, the wheat head class.

## 3 Methodology

### 3.1 Methodology

Two YOLOv5 convolutional neural networks are used. The configuration of the networks and training setup is exactly the same, as well as the set of images the networks are trained on. The difference between the networks is that for the images of one of them, an FFT-mask is applied. This network is therefore referred to as Masked-YOLO for the purposes of this paper, while the other network as Plain-YOLO. In addition, Plain-YOLO is fitted with a low-pass filter to construct a Filtered-YOLO which is used to compare the effectiveness of the FFT-masking with a simple filtering technique.

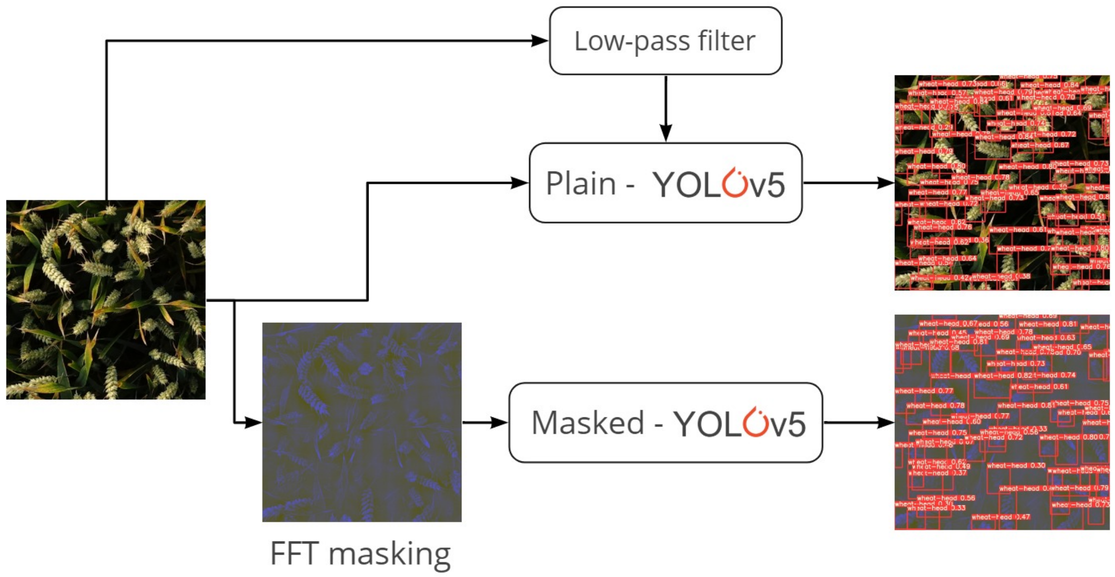


Figure 5: Graphic demonstrating of the testing pipeline. The same test set is tested three times on three different setups. Once on Plain-YOLO, once on Plain-YOLO after a low-pass filter is applied and once on Masked-YOLO.

The FFT-mask applied is the high-frequency suppression part of the method. In order to achieve this, there are two filters applied. A low-pass and a high-pass filter that combine to add up to, essentially, a band-pass filter. The threshold above and below which frequencies are removed is determined by the dataset itself. More specifically, the average frequency of the labeled data, which is in the form of bounding boxes, is calculated. Hence, frequencies above and below the average frequency of those bounding boxes are removed. This should theoretically improve adversarial performance since adversarial perturbations occur on a high-frequency spectrum. The lower frequencies are also theorized to be able to improve performance in standard accuracy since they are technically unnecessary information but such an effect is not observed in this paper. Quite the contrary, in a normal setting, a drop in performance is observed.

Measuring standard accuracy is done immediately after the networks are trained. In fact, YOLOv5 is integrated with a tool called "WANDB"[1], the purpose of which is to record an extensive analysis and multiple performance metrics for all training cycles. In contrast, to measure adversarial performance, adversarial attacks are mocked and used for separate testing.

## 3.2 Code Description

To fully utilize these tools, as well as to minimize the number of actions that are needed to be done in order to run the described experiments, a script was written in Python. This script allows the user to input the number of images they would like to use, from the more than 3.5 thousand images contained in the original dataset, the batch size, which refers to the number of training examples used per iteration of the network, and the epochs/iterations the

[1]Weights & Biases: Developer Tools for Machine Learning, https://wandb.ai/site

network goes through during training. The script then constructs a training, validation and testing set out of this network as well as prepares the labeled data to the format expected by YOLOv5 and trains an instance of the network on this data (Plain-YOLO). After training, it only stores the weights of the network. It then replaces the training and validation set with the same set of images, but only after applying the FFT-mask on them, as described above. Another instance of the network is then trained on these images (Masked-YOLO) and then, the weights of this network are also stored. Finally, adversarial attacks are mocked on different images than the ones in the training set and the resulting performance of the networks is measured. Plain-YOLO's performance is simply tested on the selected images (with and without a low-pass filter) while Masked-YOLO's is tested after the FFT-mask is applied on them (this pipeline is also described in Figure 5).

## 4 Contribution

The main contribution that this paper is examining is whether adversarial attack performance or otherwise robustness, can be improved by examining the frequency information contained within the labeled data of an image and eliminating frequencies outside of the range observed within this labeled data. This includes very high-frequency information, which could exist due to unforeseen noise in the data or adversarial attacks, or very low frequencies contained in the background of the images.

The conclusion formed after conducting the experiments described in the later sections is that analyzing frequency information and removing unnecessary frequencies can indeed improve adversarial performance. At the same time, this improvement, which adds an overhead of almost an hour when training the network, is comparable in performance with a simple low-pass filter, which has almost no overhead and is created without any frequency analysis to determine its threshold but simple trial-and-error. This means that while the FFT-mask creation method does show promise, the method for constructing the mask is not fully optimized yet.

### 4.1 Experimental work

There are two identical YoloV5 networks trained, one in normal data (Plain-YOLO) and one in data that the hypothesized "unnecessary" frequencies have been extracted from the data (Masked-YOLO, as described in section 3). Then, after training these networks and testing their performance on normal data, a test set is created where the data in this set has been adversarially perturbed using a black-box attack. This is done two different times, once for the attack constructed with "Noisify" and another for "Controlled Gaussian Noise", both explained in section 5. Lastly, the performance of the networks is once again measured on these test sets. Plain-YOLO is tested twice on each set, once with and once without a simple low-pass filter applied to the images. It is worth noting that the same FFT-mask that was used during training one of the networks is used on this test set when measuring performance.

### 4.2 Improvement of an existing idea

The main improvement of an idea in this paper is the generation of the FFT-mask based on the frequency information within the labeled data, as the title suggests. Meaning that, while there are already methods of using Fourier Transforms to decompose data to their constituent frequencies and recompose them after applying specific transformations to those frequencies, there are no techniques for using information obtained from the labeled data as a guideline to determine which frequencies to eliminate.

The results produced after implementing this idea are presented and discussed in section 5. Essentially, the FFT-mask indeed leads to an improvement in accuracy when the performance is tested on adversarial examples (robustness), as was hypothesized. This points to the fact that the FFT is very likely to be a useful approach for high-frequency suppression as an adversarial defense. At the same time, the performance improvements are almost negligible when compared to a simple low-pass filter. This leads to further proposed improvements that could be used to refine the approach explored in this paper.

### 4.3 Proposed improvements

The main "bottleneck" that currently limits the effectiveness of this approach is the way the FFT-mask is generated. Meaning that currently, bounding boxes tend to contain some unrelated information regarding the wheat heads, due to their shape.

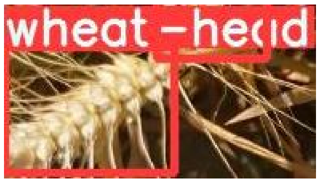


Figure 6: A typical label on the wheat head dataset contains a significant amount of background information

One can see that between the "hair" of the wheat head there is a significant amount of background information which will inevitably contribute to decreasing this average frequency calculated through the FFT. This means that one would either have to manually create labels specific for each wheat head (perhaps more closely resembling an oval shape than a rectangle) so that background information is eliminated as best as possible, or do some further analysis on the frequency spectrum determined by the FFT. The latter referring to the fact that the FFT of such labeled data would, most likely, have concentrations of frequencies around two areas. One in a relatively low-frequency spectrum and another, likely more prevalent, one in a higher spectrum. One could determine these "local optima" and attempt to extract only the highest one.

Another possible improvement also concerns the FFT-mask configuration. As has been previously stated adversarial attacks tend to occur in a higher frequency spectrum than most everyday objects/data. Hence, one could reverse the FFT-mask generation and instead of analyzing the data itself to determine the frequency setting, they could analyze the adversarial perturbations. Referring back to Figure 1, one could extract the "noise" as is displayed in the second image and construct a band-reject filter to be used for the data.

## 5 Experimental setup, hyper-parameters and results

In order to assist with reproducibility of this research, as well as transparency of the experimental process, the specifications of the computer that the networks were trained and tested on are included below.

- **Graphical Processing Unit (GPU)**: Nvidia GeForce GTX 970, 4GB
- **Central Processing Unit (CPU)**: Intel(R) Core(TM) i5-4430 CPU @ 3.00GHz
- **Random-access memory**: 16GB
- **Operating System**: Windows 10

In addition, there are multiple hyperparameters of the YoloV5 network and the FFT-mask that were tuned according to the specifications of the problem and the computer.

- **YoloV5**
  - **Number of images used from the dataset (per iteration)**: 1000, 70% used for training, 20% for validation and 10% for testing.
  - **Batch size (for training & testing)**: 2
  - **Epochs**: 50. Some training sessions were performed with 100 epochs but there was no significant performance improvement after 50.
  - **Weights**: yolov5s.pt. The YoloV5 network has multiple initial weights that can be used as a basis for training. The yolov5s (small) weight configuration was used since it is the fastest one.
  - **Memory settings**: no weights between iterations were saved. Only the last iteration and the training and testing datasets were fully cached.
  - **Intersection Over Union (IOU)**: 0.7. This is a setting that is only related to the validation and testing parts. Since the network is identifying wheat heads by drawing bounding boxes around them, it very rare that the predicted bounding box is exactly on top of the label. IOU solves this by allowing one to set a "similarity" metric which compares the common area between the bounding box of the ground truth and the predicted one (area of overlap/area of the union) [9].
- **Fast-Fourier Transform**: The way the FFT was used in this paper is essentially a combination of a low-pass and a high-pass filter. These filters were optimized to retain frequencies close to the spectrum of the wheat heads and discard the rest.
  - **Low-pass filter**: 120
  - **High-pass filter**: 0.2
- **Noisify**: Python package used to introduce noise in any type of data. The two types of noise parameters mentioned below refer to pre-existing implementations of noise within the package [4].
  - **Human-error**: Set to 20. Represents attribute scrambling and random noise.
  - **Machine-error**: Also set to 20. Represents Gaussian noise and occasional interruptions to the input.
- **Controlled Gaussian Noise**: Numpy and SciPy were used to construct Gaussian noise (on a low, high and the full frequency spectrum of the images) as well as to make a filter that removes high frequencies from the images.
  - **NumPy Noise**: The numpy.random.rand() method was used with the image array as input. The resulting values were amplified by a factor of 40.

– **Scipy Filter**: The ndimage.gaussian_filter() method was used from SciPy. This method was used to filter out some of the frequencies in the noise to vary the black box attacks as well as to create a low-pass filter for Filtered-YOLO. The former has a sigma of (5, 5, 0) while the latter a sigma of (1.5, 1.5, 0).

Before presenting the results it is worthwhile to mention that, while there is no one way to measure the performance of a neural network, these are the main metrics used to do so.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AveP(q)}}{Q}$$

$TP$ = True positive

$TN$ = True negative

$FP$ = False positive

$FN$ = False negative

Figure 7: Equations for calculating precision, recall and mAP based on predictions [9]

- **Precision** refers to the percentage of correct predictions from the predicted wheat heads.
- **Recall** refers to the percentage of examples that were properly predicted compared to the sum of the labeled wheat heads.
- **mAP**: is referring to the *mean average precision* which is a combination of different precision measurements by varying the IoU configuration (discussed in section 2) over Q, which is the number of queries.

Before the performance evaluations, the network are trained with the hyperparameters mentioned above. To show that both of the networks are properly trained and overfitting is avoided the training and vaildation loss curves are included below.

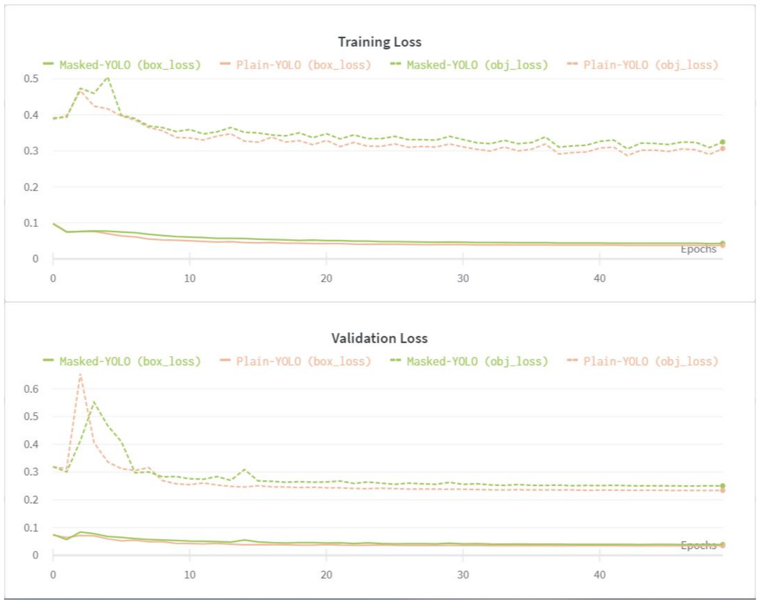


Figure 8: Both curves converge but epochs are limited to avoid overfitting

The above networks produce the following results:

| Performance Metric | Plain-YOLO | Masked-YOLO | Filtered-YOLO |
|---|---|---|---|
| **Precision** | 0.9306 | 0.9118 | 0.916 |
| **Recall** | 0.8939 | 0.8475 | 0.807 |
| **mAP@0.5** | 0.9435 | 0.9082 | 0.883 |
| **mAP@0.5:0.95** | 0.5137 | 0.4614 | 0.468 |

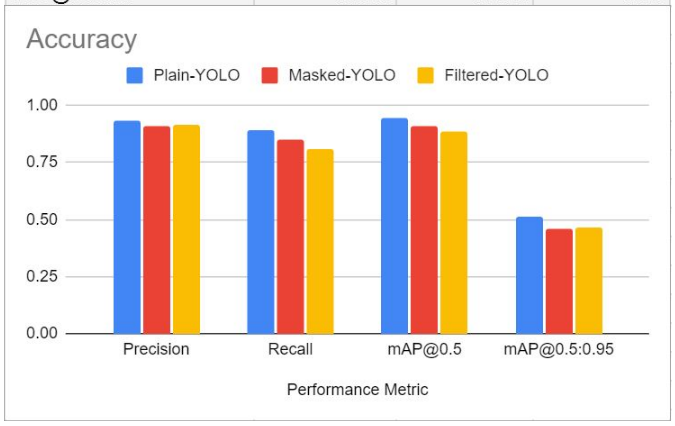


Figure 9: Masked-YOLO and Filtered-YOLO perform worse than Plain-YOLO

For all of the metrics, a higher value means better performance. There are some examples where the Masked-YOLO indeed has improved precision and recall by a slight margin but the example displayed above is representative of most runs. Hence, the FFT-mask seems to be worsening instead of improving performance. This is against the initial hypothesis that the FFT-mask could also improve accuracy due to filtering out unnecessary information but does not convey any information about the robustness of the models yet. What can be determined from the graph is that the FFT-mask implemented for Masked-YOLO, even when constructed as precisely and tailored to the wheat head class as it currently is, still removes information that the network deems relevant and uses to form features that help with classification.

In order to test the robustness of the models, we are performing two black-box adversarial attacks (as defined in section 2). Both attacks are designed for the purposes of this paper and involve inserting high-frequency noise in the pictures. One does so using a python package called "Noisify" [4] while the other uses the python library "SciPy" [16]. An example of such an adversarial perturbation for both approaches is displayed below.

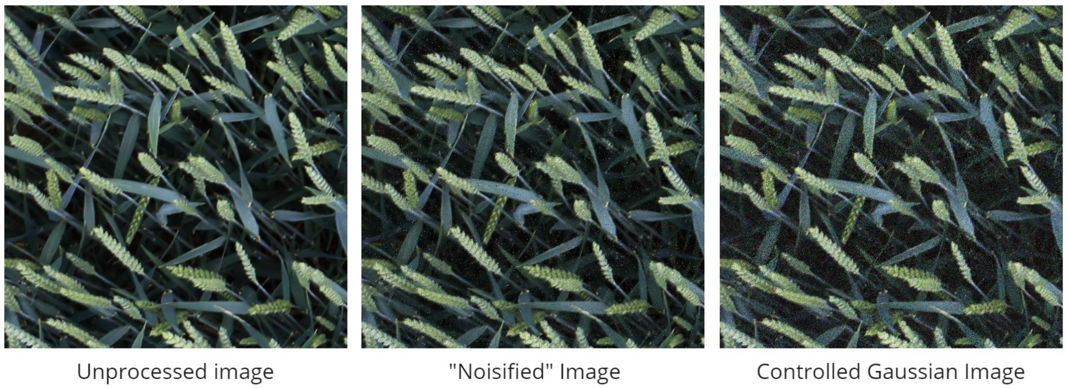
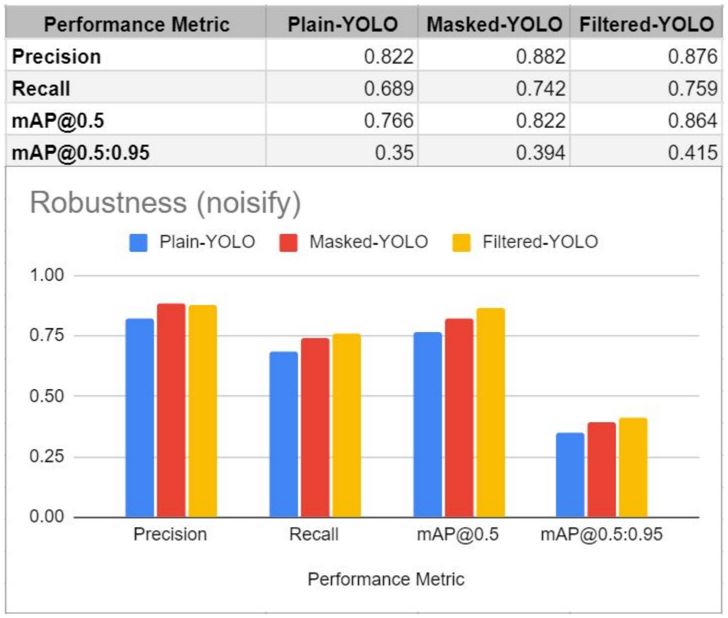


Figure 10: Example of the black box attacks applied. The noise is visible but negligible for the human eye.

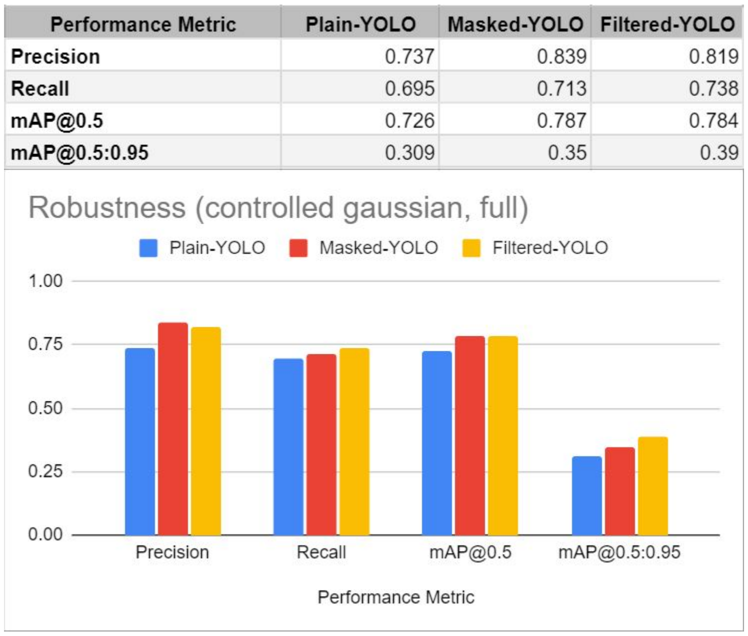
As is evident from the pictures, while for the human eye the wheat heads are still almost just as easily distinguishable, this is not the case for the YoloV5 model which has a lot of trouble classifying the adversarially perturbed image.

Running the same networks as before on these transformed sets of images in order to test robustness produces the following results.



| Performance Metric | Plain-YOLO | Masked-YOLO | Filtered-YOLO |
|---|---|---|---|
| Precision | 0.822 | 0.882 | 0.876 |
| Recall | 0.689 | 0.742 | 0.759 |
| mAP@0.5 | 0.766 | 0.822 | 0.864 |
| mAP@0.5:0.95 | 0.35 | 0.394 | 0.415 |

Figure 11: Masked-YOLO and Filtered-YOLO both perform better than Plain-YOLO on Noisify attacks



| Performance Metric | Plain-YOLO | Masked-YOLO | Filtered-YOLO |
|---|---|---|---|
| Precision | 0.737 | 0.839 | 0.819 |
| Recall | 0.695 | 0.713 | 0.738 |
| mAP@0.5 | 0.726 | 0.787 | 0.784 |
| mAP@0.5:0.95 | 0.309 | 0.35 | 0.39 |

Figure 12: Masked-YOLO and Filtered-YOLO both perform better than Plain-YOLO on Controlled Gaussian attacks

Unlike the previous chart, this time the Masked-YOLO and Filtered-YOLO are actually performing better than Plain-YOLO by a significant margin. This means that even though the hypothesis that the FFT-mask could produce accuracy improvements was not proven to be true for this setting, it did manage to successfully suppress high-frequency components and effectively improve the robustness of the masked model. This is an indication that the FFT truly can provide some useful insight for a neural network by limiting unnecessary components, especially on noisy images. At the same time, the performance of Masked-YOLO is comparable with the simpler, Filtered-YOLO. This is unfortunate since it takes up to one hour for the FFT-mask to be created while filtering the images only takes seconds.

To conclude, the FFT-mask did prove to be successful in increasing adversarial performance but not in a cost-effective way. It is still an interesting approach that, if optimized, could potentially bring improvements both in the accuracy and the robustness evaluation.

## 6 Responsible Research

Scientific inquiry requires more than a simple statement of the facts discovered and their interpretation. It calls for the researcher to be responsible for their actions, especially when it comes to honestly and thoroughly recording their process so that it remains reproducible in the future.

In recent years, neural networks have become notoriously difficult to analyze and explain their behavior. The discovery of adversarial attacks in itself is an indication that we do not necessarily understand how neural networks work and the intricacies behind their decision-making process. We can evaluate their performance overall but explaining case-by-case scenarios and classifications is a problem the scientific community is still tackling with [11]. Similarly, reproducibility is also a complex area when it comes to neural networks. For this paper, we are using the open-source implementation of YoloV5 [15] as well as the open-source database of wheat head images from the Kaggle competition [6]. The details of the experimental/computing setup are described in section 5. Furthermore, during the training of the networks, there were no other techniques introduced that were observed to improve performance for this dataset, (such as pseudo labeling[2]) since those techniques can introduce additional "overhead" when it comes to reproducibility. This is due to the fact that there might be multiple configurations with which those improvements can be implemented. There were also multiple training iterations and experiments performed so that the data is not limited to specific, selected iterations. Lastly, the selection of the subset of images that were used during training, validating and testing was randomized per iteration in order to avoid results that would be skewed in relation to those images.

Theoretically speaking, with all the information provided above one should be able to now reproduce the findings to a very high degree of accuracy, with the randomization factor of the images only affecting the results. Practically speaking, neural networks end up introducing more factors affecting reproducibility. Even if one chose the same exact subset of images, the translation and transformation of the initial weights of the network (yolov5s.pt) are randomized per epoch. This effectively leads to different values for metrics like precision and recall when the final network is evaluated. This randomization could technically be reproduced by extracting the seed used by YoloV5 but this would defeat the purpose of having results that can generalize to convolutional neural networks. Once again, this is related to the nature of neural networks themselves and not due to an error in the method.

[2]https://www.kaggle.com/orkatz2/ensemble-yolov5-pseudo-labeling-d5-d7-resnest

# 7 Conclusions, Discussion and Future Work

## 7.1 Conclusions

To summarize the answer to the research question of this paper. Indeed, we can use object-specific frequency information from labeled data to improve a CNN's robustness to adversarial attacks as is seen in Figure 11. We just can not do so without any drawbacks. The main ones being:

- **Accuracy/Robustness trade-off**: It seems that there is no clear way to simultaneously improve standard accuracy and robustness. As was explored by Zhang H. et. al. [18], there is a "Theoretically Principled Trade-off between Robustness and Accuracy" which was proven to be true in the case of this paper since the FFT was not able to suppress high and low frequencies in such a way as to not remove any information crucial to object detection, leading to a decrease in standard accuracy as seen in Figure 9. Even when designing the FFT-mask based on the labeled data (as described in section 3) and very carefully excluding frequencies, the trade-off still holds.
- **Efficiency**: In order to design the FFT-mask, the whole of the training set is sampled and the average frequencies of the bounding boxes are used. This process takes a significant amount of time for a training set of 1000 images (about 1 hour). Training the models also takes about 1.5 hours which means that the training time for the Masked-YOLO is almost doubled. Thankfully, for later object detection the Masked-YOLO can simply apply the already created mask on the images so the effect is expected to be more limited during actual use for detection but even then, it is not insignificant. It takes a few seconds to apply the FFT-mask on a single image which would have a measurable effect on the performance of YOLO if used on a video for example. (YOLO can currently process "an astounding 155 frames per second" [15], abstract).

The results produced in this report are not necessarily new or surprising when compared to the state-of-the-art research that is currently being conducted regarding adversarial defenses. Suppressing high-frequency components is a well-known technique that can be used to improve robustness, while Zhang, Z et. al. [19] actually used the Discrete Fourier Transform in their own paper. It was the strategy that, used individually, produced the highest robustness score when compared to adversarial training and model ensemble [19].

The new element introduced in this paper is the additional analysis of the labeled data. Zhang, Z et. al. [19] used frequency suppression and the Discrete Fourier Transform similarly, by determining a threshold through trial-and-error, while in this paper the threshold is determined by the labeled data. This is important because this means that this method generalizes to any object detection setting (like the wheat head dataset). Also, in Zhang's method, the threshold is optimized based on performance in a specific adversarial setting but often there are no adversarial examples to select and cater to when utilizing an adversarial defense strategy. As mentioned in section 2, there are multiple ways of conducting adversarial attacks with new ones still being discovered. The method proposed in this paper partially "defends" against any such attacks that focus on manipulating high-frequency information within the image, without needing any adversarial data (unlike Zhang, Z et. al. [19] and methods such as adversarial training, mentioned in section 2.2).

## 7.2 Discussions and Future Work

There is an array of possible improvements one could implement to properly and more exhaustively address the research question. While this paper was able to provide a positive and definitive answer to it, the method used is still largely experimental and unexplored.

First of all, the method for determining the FFT-mask is sub-optimal. This is elaborated on further in section 4.2. The proposed improvements contained in this section might not necessarily be the most optimal way to determine the mask but they would likely produce a much more accurate mask which could lead to a lesser drop in standard accuracy and/or further improvements when it comes to robustness. It could even be the case that a highly precise mask construction method will be the first adversarial defense that does not necessarily entail a significant drop in standard accuracy, if not an increase. Although both the former and the latter are simply a hypothesis.

Then, there is the selection of the network, YoloV5. The selection of this convolutional neural network was simply based on the performance evaluations of the Kaggle competition that inspired this paper [6]. There are, possibly, other classifiers/object detectors that could perform much better when combined with the Fast Fourier Transform. This is due to the fact that YOLOv5 already performs a substantial amount of post-processing on the data it is trained and it later detects. This could, in theory, be clashing with the FFT, since it is also essentially a method of post-processing. One could either try different networks with as similar hyperparameters as possible, or even explore different ways of integrating YoloV5 with the FFT. Perhaps the latter could be done by introducing the FFT as a post-processing step in an inner layer and not simply on the data provided as input to the network.

Lastly, the adversarial "attack" performed on this paper is relatively limited. A black-box adversarial attack that does not utilize any of the properties of the network can hardly be called an "attack", even though by definition it is one. It more closely resembles noise that could be caused by malfunctioning devices in the data collection process or contained in corrupted data. In any case, this is interesting because the increase in performance when evaluating Masked-YOLO on this noise means that one could also use the Fast Fourier Transform to construct some type of "denoiser" for networks whose data is often noisy and corrupted. At the same time, in order to exhaustively evaluate the research question, multiple types of adversarial attacks should be introduced.

# A Appendix

Included below are two graphs that were not necessary for the conclusion of the paper to be formed but are still interesting when it comes to examining the performance of the networks under different situations.

| Performance Metric | Plain-YOLO | Masked-YOLO |
|---|---|---|
| **Precision** | 0.752 | 0.847 |
| **Recall** | 0.664 | 0.727 |
| **mAP@0.5** | 0.719 | 0.797 |
| **mAP@0.5:0.95** | 0.32 | 0.371 |

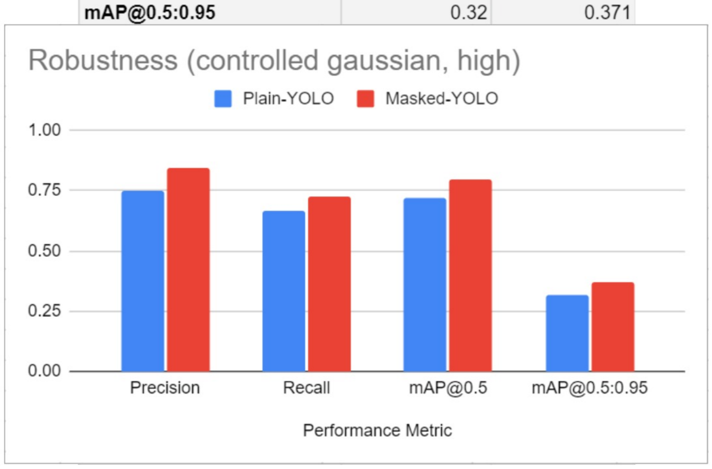


Figure 13: Masked-YOLO performs better on high frequency noise

| Performance Metric | Plain-YOLO | Masked-YOLO |
|---|---|---|
| **Precision** | 0.911 | 0.893 |
| **Recall** | 0.875 | 0.844 |
| **mAP@0.5** | 0.921 | 0.889 |
| **mAP@0.5:0.95** | 0.483 | 0.458 |

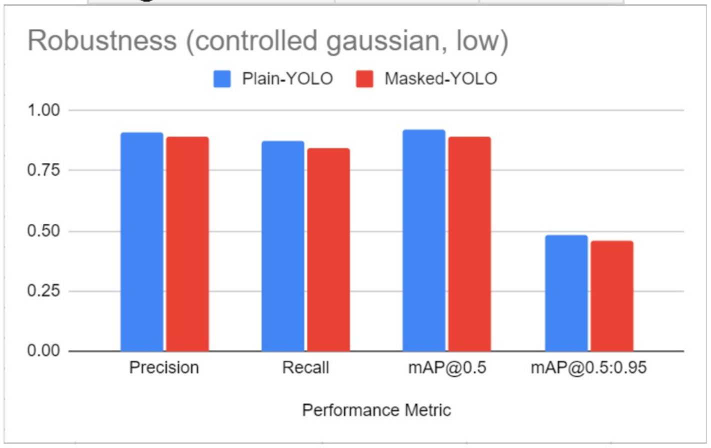


Figure 14: Masked-YOLO performs worse on low frequency noise

The graphs above display an interesting property of the FFT-mask. While it is still increasing performance for high-frequency noise, the same is not true for low-frequency noise. The mask fails to eliminate background information only and, most probably also eliminates information crucial to wheat head detection.

## References

[1] Balsys, R. (2021). *Yolo v3 with TensorFlow 2. (Python Lessons).* https://pylessons.com/YOLOv3-TF2-mnist/

[2] Bochkovskiy, A. et al. (2020) *YOLOv4: Optimal Speed and Accuracy of Object Detection. (Computer Vision and Pattern Recognition).* Published. https://arxiv.org/abs/2004.10934, p.2

[3] Dong, Y. (2017, October 17). *Boosting Adversarial Attacks with Momentum. (Machine Learning)* https://arxiv.org/abs/1710.06081

[4] D. (2019). *Noisify. GitHub.* https://github.com/dstl/Noisify

[5] Engstrom L. et al. (2018). *A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. (ICLR 2019 Conference)* https://openreview.net/forum?id=BJfvknCqFQ, abstract

[6] *Global Wheat Detection: "Can you help identify wheat heads using image analysis?"* (2020, September). Kaggle. https://www.kaggle.com/c/global-wheat-detection/overview

[7] Goodfellow I.J. et al. (2014). *Explaining and Harnessing Adversarial Examples. (Machine Learning)* Published. https://arxiv.org/abs/1412.6572, p.1

[8] Hamidi, R. (2019, December 2). *Deep Learning Application in Time-Frequency Analysis for Noise Attenuation | Earthdoc.* https://www.earthdoc.org/content/papers/10.3997/2214-4609.201977037, abstract

[9] Hui, J. (2020). *mAP (mean Average Precision) for Object Detection. (Medium).* https://pylessons.com/YOLOv3-TF2-mnist/

[10] Ilyas, A et al. (2019). *Adversarial Examples Are Not Bugs, They Are Features. (Machine Learning)* https://arxiv.org/abs/1905.02175, abstract

[11] Keane T.M., Kenny M.E. (2019). *How Case Based Reasoning Explained Neural Networks: An XAI Survey of Post-Hoc Explanation-by-Example in ANN-CBR Twins. (Artificial Intelligence)* https://arxiv.org/abs/1905.07186

[12] Madry A. et al. (2017). *Towards Deep Learning Models Resistant to Adversarial Attacks. (Machine Learning)* Published. https://arxiv.org/abs/1706.06083, p.2

[13] *Overview of model structure about YOLOv5 #280 (GitHub Repository).* https://github.com/ultralytics/yolov5/issues/280

[14] Rathore, P. (2021). *Untargeted, Targeted and Universal Adversarial Attacks and Defenses on Time Series. (Machine Learning)* https://arxiv.org/abs/2101.05639, p.1

[15] Redmon, J., Divalla, S., Girshick, R. & Farhadi, A. (2016) *You Only Look Once: Unified, Real-Time Object Detection. (Computer Vision and Pattern Recognition).* Published. https://arxiv.org/abs/1506.02640

[16] SciPy Developers (2021). *SciPy library. SciPy.Org.* https://www.scipy.org/scipylib/index.html

[17] Yao, S. (2019, February 21). *STFNets: Learning Sensing Signals from the Time-Frequency Perspective with Short-Time Fourier Neural Networks (Machine Learning)* https://arxiv.org/abs/1902.07849, abstract

[18] Zhang, H. et al. (2019). *Theoretically Principled Trade-off between Robustness and Accuracy. (Machine Learning)* Published. https://arxiv.org/abs/1901.08573

[19] Zhang, Z., Jung, C. & Liang, X. (2019) *Adversarial Defense by Suppressing High-frequency Components. (Computer Vision and Pattern Recognition)* Published. https://arxiv.org/abs/1908.06566, p.3