AxFuncta

Fit instead of Predict for Accidental Scene Conditions

Pattern Recognition and Bioinformatics

Vladimir Rullens

Delft University of Technology

TU Delft

AxFuncta

Fit instead of Predict for Accidental Scene Conditions

by

Vladimir Rullens

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Monday July 6, 2026 at 11:00 AM.

Student number:	5329779	
Project duration:	November 17, 2025 – July 06, 2026	
Thesis committee:	Dr. J. C. van Gemert,	TU Delft, Thesis Advisor, Chair
	Ir. A. Gielisse,	TU Delft, Daily Co-Supervisor, Advisor
	Dr. R. Marroquim,	TU Delft, Core Member

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Preface

Presented here is my master's thesis titled "AxFuncta: Fit instead of Predict for Accidental Scene Conditions", written for the MSc Data Science and Artificial Intelligence Technology program at the Technical University of Delft. This project has spanned from November 2025 to the start of July 2026, during which I was actively engaged in my research and the writing of this thesis.

I want to greatly thank my supervisors, Dr. Jan van Gemert and Ir. Alexander Gielisse, for all the feedback they have given me, and for providing me the opportunity to perform and complete this project. I especially want to thank Ir. Alexander Gielisse, for being my co-supervisor, and for providing incredible insight during our meetings, which always allowed me to move forward during this project. His incredibly clear explanations alongside his great wisdom have become an inspiration that Ill take with me in the future. I also want to thank Dr. Jan van Gemert, for being my supervisor and chair, and for the discussions that steered this project in the right direction. Furthermore, I would also like to thank Dr. Ricardo Marroquim, for accepting to be a core member of the committee, and for the in-depth feedback that he gave near the end of this project.

Finally, I want to thank my friends and family, who have supported me during this endeavor. Even if I was never truly able to convey my work to most of them, they were always willing to listen and nudge me forward.

Vladimir Rullens
Delft, June 2026

Contents

Preface i

1 Introduction 1

2 Preliminary 2
- 2.1 Neural networks . . . 2
  - 2.1.1 Training . . . 3
  - 2.1.2 Convolutional Neural Networks . . . 4
  - 2.1.3 Task types . . . 5
- 2.2 Image transformations . . . 5
- 2.3 Combating Geometric variation . . . 7
  - 2.3.1 Spatial Transformer Network . . . 7
  - 2.3.2 Deformable Convolutional Network . . . 8
  - 2.3.3 Group Equivariant Convolutional Networks . . . 8
  - 2.3.4 Canonicalization . . . 8
- 2.4 Image Congealing . . . 9
- 2.5 Data augmentation . . . 10
- 2.6 Main architecture pre-requisites . . . 10
  - 2.6.1 Model-Agnostic Meta-Learning . . . 10
  - 2.6.2 Feature-wise Linear Modulations . . . 11
- 2.7 Implicit Neural Representations . . . 12
- 2.8 From Data to Functa . . . 13
- 2.9 Additional terms . . . 14
- 2.10 Datasets used . . . 14

References 16

3 Scientific Paper 18

4 AI Statement 32

1

Introduction

The field of computer vision has been researched greatly, bringing us machine learning models that can accurately classify the main object present in a high-quality, real-world image, or even generate completely new images and videos from a text prompt. However, these models are entirely dependent on the distribution of the data that they learn from. As an example, if a model has only seen images of sitting cats, then it will be unaware of the fact that they are still cats even if they lie down on their sides. This in turn results in e.g. a misclassification. Because of this, models require significant amounts of data to become aware of all the possibilities that exist in the real world. Even worse, if this quantity of data is not available in a given setting, then the model will fail to be robust to unseen situations.

This paper explores an option that closely resembles an idea that has been abandoned in favor of newer machine learning strategies, namely the act of fitting unseen data into a known distribution. In the case of our cat example, this would be the act of rotating our lying cat until it looks like it is sitting. This allows us to greatly reduce data requirements, while making models more robust to unexpected situations and use cases. An attempt to revitalize this idea is performed by utilizing a more recent work within the field of computer vision focused on reconstructing images.

This report is split into two parts. Following this introduction, part 2 will cover the relevant computer vision background that is part of our research. After this, part 3 will cover our performed research in the form of a research paper. The paper describes the existing work more broadly, covers our method, and showcases experimental results comparing our method to existing alternatives. An alternative idea that was abandoned in favor of our final method is also showcased in its appendix. Finally, a short AI Statement is posted in part 4.

2

Preliminary

2.1. Neural networks

The field of computer vision is built on the existence of neural networks. At their core, these neural networks are based on mathematical models that take input values and learn to match those as closely as possible to their respective output values, such as matching the color of an apple to its ripeness. At its simplest, this can be the following line, where x represents the input, modified by a slope w and a bias b, to obtain the output y:

$$y = xw + b \tag{2.1}$$

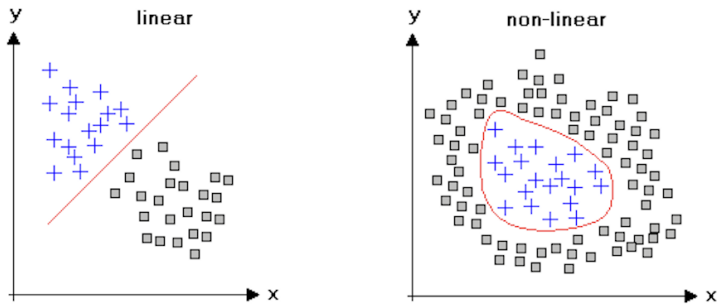

Figure 2.1: An example of decision boundaries splitting two classes into their own regions. Left: A dataset that is cleanly separable using linear functions. Right: A dataset where non-linearity is required to split both classes. Image from [25]

One can then either fit this line as closely as possible to a set of datapoints, or form a **decision boundary** that separates two classes. Unfortunately this format has limited expressive power, only being able to represent linear functions. An example of this issue is shown in Figure 2.1. While the situation on the left allows for clean separation of both classes using a linear function, this is not possible for the situation on the right.

To fix this, neurons, also commonly referred to as perceptrons, were introduced [17]. These take a set of input values $X = [x_0, x_1, ..., x_n]$, and map these to an individual output value y through a set of **weights** $W = [w_0, w_1, ..., w_n]$ and a **bias** term b, creating the following function:

$$y = \sigma(\sum_{i=0}^{n} x_i * w_i + b) \tag{2.2}$$

Here, σ represents a non-linear activation function, enabling the level of expressivity that we were looking for proper separation of data. Commonly used activation functions are the Rectified Linear Unit

(ReLU), which sets all negative values to 0, as well as the Hyperbolic Tangent $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, which maps all values between negative and positive infinity to the range [-1, 1].

Using this neuron as a building block we can now create our neural networks. To this end Figure 2.2 showcases an example neural network comprised of an input layer, a single hidden layer, and an output layer, built using multiple neurons. Using this network we can perform many types of tasks such as the **classification** of data into different categories.

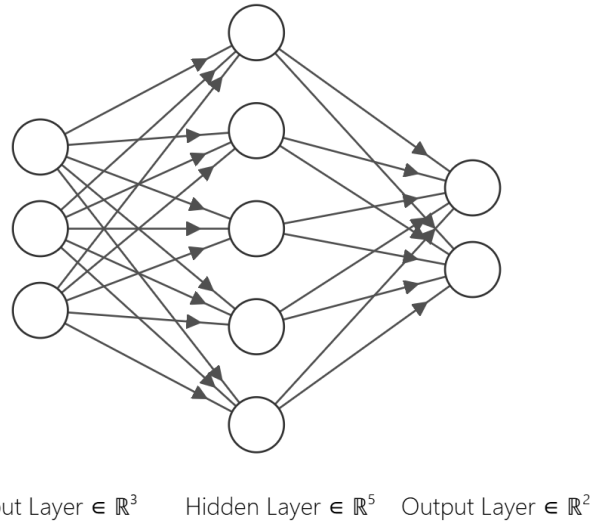

Figure 2.2: A standard neural network, containing an input layer, a single hidden layer, and an output layer. These layers contain 3, 5, and 2 neurons respectively.

When working with great amounts of complex data the size of the above network may grow exceedingly large, potentially containing billions of weights and biases to determine what object is present within a high-resolution image. In turn, finding the optimal value for each of these becomes non-trivial. To this end, training is performed.

2.1.1. Training

Training is the process of ensuring that, given some input, our output matches what our dataset expects as closely as possible. For this example, our task will be 2-class classification, where the top neuron in Figure 2.2 states the probability of our apple being immature, and the bottom neuron states the probability of our apple being ripe. Let's say we take some input values from our dataset, such as the color of 8 different apples in the rgb color space. Initially, the weights and biases of our network are randomly initialized. As such, when performing the above-mentioned Equation 2.2 for each neuron, titled a **Feed-Forward pass**, the output will similarly be random and therefore may be wildly inaccurate. To determine the level of inaccuracy between our obtained output and our expected output, a **loss function** is utilized. For classification, this tends to be the **Cross-Entropy Loss** function shown as follows:

$$CELoss = -\frac{1}{N}\sum_{i=0}^{N}\sum_{c=0}^{C} y_{ic}\log(\hat{y}_{ic}) \tag{2.3}$$

Here, y represents the expected probability for our class, with $\hat{y}$ representing our prediction. In the case of ripe apples, the expected probabilities would be [0.0, 1.0], indicating [probability immature, probability ripe]. This is then summed over each class c and averaged over each apple i part of a batch, allowing us to perform the above computation in parallel for multiple apples part of our dataset at a time. We refer to N as the **Batch Size**. Using this loss function on our prediction gives us our **Loss** value, which should be minimized in such a way that not only the training samples, but also unseen samples from a **Validation** or **Test** set obtain a loss value close to 0.0. To obtain a lower loss value, and therefore get a closer to accurate prediction, we perform **backpropagation**.

Backpropagation is the act of updating the weights and biases of our model based on the loss value that we obtained from passing training data through our network. The base method for this is called **Stochastic Gradient Descent** (SGD). Namely, this method utilizes the chain rule to compute the gradient for each weight and bias value, working from the loss value back to the input. By following this gradient in the opposite direction, we can update the weight and bias values such that the current samples obtain better results. The **Learning Rate** η then determines how much the weight and bias should be moved by, resulting in the following equation, where ρ indicates the learning rate, and θ indicates our parameters i.e. its weights and biases.

$$\theta = \theta - \eta \frac{\delta(Loss)}{\delta(\theta)} \tag{2.4}$$

While training, this learning rate should be tuned on the previously discussed validation set such that our model learns to generalize across the entire training distribution. If the learning rate is too low, the gradients may get stuck in local minima, preventing accurate classification, whereas if the learning rate is too high, the model may overshoot optimality. There are also improved versions of SGD, such as the **Adam optimizer** [11], which smoothes gradients by incorporating gradient history from previous passes into the current gradient, making it more robust to inconsistent (noisy) gradients and allowing for faster training.

Finally, tricks can be applied that further optimize the performance of our model. One such method is **L2 regularization**, expressed as the following formula:

$$\text{L2} = \sum_{i=0}^{N} \theta_i \tag{2.5}$$

By utilizing this formula on the weights and biases of our neural network, and incorporating it into our loss function, we can limit the strength of the action that each neuron can perform. This allows us to prevent our models from **overfitting** onto our training set, instead forcing it to prioritize generalization to unseen instances.

2.1.2. Convolutional Neural Networks

While neural networks are powerful, the version described so far requires input data to be placed in 1D vectors. As such, a level of creativity needs to be utilized to make use of them with data types that are not natively in this format, such as images, which are a 2D data type. A naive approach to performing operations on images would be to flatten the 2D image format into a 1D vector, but as a result, spatial relationships between pixels and color channels would be lost. On top of this, images are large, containing many individual pixels. Having to process each pixel individually through a neural network would require a significant number of parameters. To address these issues, Convolutional Neural Networks (CNNs) are used [13].

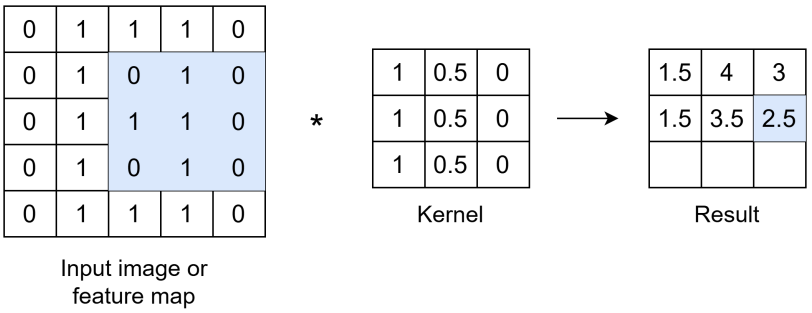

Figure 2.3: The workings of a CNN layer. Here, a global 3x3 kernel performs a neural network-style computation while taking into account neighboring pixels. The depicted image shows the operation being performed on a single image layer, where an individual multiplication operation is colored.

Figure 2.3 showcases an example of the type of computation that a CNN layer performs. A CNN layer makes use of sets of kernels that are of size $n * n$, where each kernel is shared across an entire color channel. By sliding this kernel over the entire channel, and then combining the result of each channel together, information from local neighbors can be combined into a singular value through element-wise multiplication. The full output of this operation, a **feature map**, is then able to represent e.g. edges or patterns. By default, this would end up shrinking the image, as is shown in Figure 2.3. However, this can be prevented by performing padding, which is the act of surrounding our input image by additional 0s. By combining these kernels with non-linear operations such as Max pooling, which passes the largest value in a region, as well as previously seen activation functions, complex derivations can be performed on images.

By utilizing a fixed kernel for a full image, the number of parameters that are required for processing an image can be greatly reduced. On top of this, a fixed kernel allows us to obtain similar results even if the object in the image is moved to a different location, as e.g. its edges will still be detected, though the level of robustness this brings is more limited than one may initially expect.

2.1.3. Task types

By using the CNN, many types of tasks can be performed within the field of Computer Vision. While each of these tasks are built on different loss functions, as well as different datasets, each one can be performed using the same neural network building blocks described above.

Regression One of the existing base tasks is regression, which operates similarly to classification. Given a set of input values $X = [x_0, x_1, ..., x_n]$, the goal is to match their respective set of output values $Y = [y_0, y_1, ..., y_n]$ as closely as possible. However, even for this simple task there are many loss functions through which the similarity of our output values to our prediction can be evaluated, each with their own pros and cons. Most commonly however, the Mean Squared Error loss function is used, depicted as follows:

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(\hat{y_i} - y_i)^2 \tag{2.6}$$

Here, $\hat{y}$ is the predicted output value that should match y. Another commonly used loss function includes the Mean Absolute Error, which removes the squaring operation. As a result, outliers have less impact on the loss function, which can provide a benefit in situations where a lot of noise is present.

Classification Another base task is classification, which has already had an introduction so far. Given a set of input values $X = [x_0, x_1, ..., x_n]$, the goal is to predict its corresponding **class**, such as the earlier example of whether an apple is ripe or immature. However, having multiple different classes represented through a single number would result in unequal distances between them, making the optimization landscape suboptimal. As such, the standard way of representing the classification problem is to use **one-hot encoding**. Namely, by giving each possible class its own corresponding value to optimize, we can focus on a set of output values $Y = [y_0, y_1, ..., y_m]$, similar to regression, where m represents the number of classes. This in turn allows us to utilize the previously mentioned Cross-Entropy Loss, where the correct prediction becomes the output vector where the correct class is equal to 1, while all other classes are equal to 0.

2.2. Image transformations

Images have become ingrained into our society. We use them to show people where we last went on vacation, or to investigate the existence of a potential tumor in a patient. However, an image may not always be in the pose that we want. A common example is found when taking pictures with a phone, where the resulting image may be rotated 90 degrees from what we expected. Rather than being forced to retake the picture however, we can instead choose to **transform** it, bringing it the exact pose that we want. Possible image transformations, namely those that are the focus of this research, include:

- **Translation** - The movement of an image across the x and y axis, representing horizontal and vertical movement respectively. This allows us to move the main object in a scene e.g. from the corner to the center of the image.

- **Rotation** - Rotating the scene around the center. This allows us to move the main object into an upright state.
- **Scaling** - Changing the size of the image while keeping its resolution the same. This allows us to zoom in on some object in a scene, or make the scene appear smaller.
- **Shearing** - A complex operation, where the x axis is scaled depending on its position along the y axis, or vice versa. This enables perspective warps, allowing us to view an object, such as a door, as if we were right in front of it, even if the image was taken from its side.

We represent the combination of these operations as a 3x3 **affine** matrix. We can then multiply this matrix with all our pixel coordinates, represented as (x,y,1) vectors. In turn, this enables us to predictably 'move' them to a new location. To make this work, we perform backward sampling: For every pixel in our target (transformed) image, we grab the color corresponding to its original location in the original image. However, this original location may not be the direct center of a pixel, landing between multiple pixels instead. In this case, rather than opting for the nearest pixel to copy, we can instead choose to incorporate the information from all its direct neighbors. The most commonly used approach for this is **bilinear interpolation**.

The general idea of bilinear interpolation is: The closer the coordinate is to the pixel, the more the pixel contributes to the final value. To do so, take the coordinates of the four neighboring points: $(x_0, y_0), (x_1, y_0), (x_0, y_1), (x_1, y_1)$, with their corresponding values $C_{00}, C_{10}, C_{01}, C_{11}$. First, we perform **linear interpolation** for the two pairs of horizontally neighboring points: $(x_0, y_0), (x_1, y_0)$ and $(x_0, y_1), (x_1, y_1)$. As an example, we perform the following equation for the first pair, where (x_p, y_p) are the current transformed coordinates:

$$I_0 = \frac{(x_1 - x_p)}{(x_1 - x_0)} C_{00} + \frac{(x_p - x_0)}{(x_1 - x_0)} C_{01} \tag{2.7}$$

$$I_1 = \frac{(x_1 - x_p)}{(x_1 - x_0)} C_{10} + \frac{(x_p - x_0)}{(x_1 - x_0)} C_{11} \tag{2.8}$$

Then, by applying linear interpolation across the y channel using our resulting values, we get our final value:

$$f(x_p, y_p) = \frac{(y_1 - y_p)}{(y_1 - y_0)} I_0 + \frac{(y_p - y_0)}{(y_1 - y_0)} I_1 \tag{2.9}$$

While this approach brings about convincing results in image transformations, its results are lacking for machine learning optimization. Using this approach, machine learning gradients that inform us of how we should update our model are only being passed through the 'direct neighbors' of pixels. This limits their ability to make informed decisions about large transformations.

In 2019, a paper titled "Linearized Multi-Sampling for Differentiable Image Transformation" by Jiang et al. introduced a variation of this method, called **linearized sampling** [9], which focuses on addressing this exact issue. An example of this process is shown in Figure 2.4.

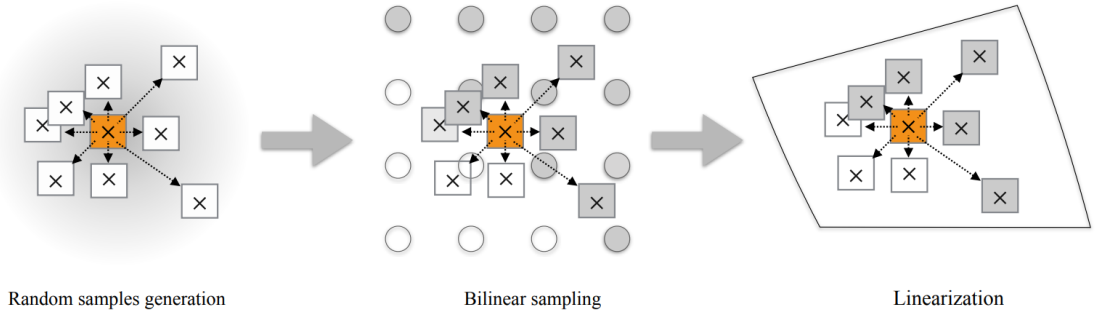

Figure 2.4: The linearized multi-sampling approach. Image from [9]

Rather than sampling directly at the current pixel location, this operation creates n dummy pixels in a range r around the current pixel location. By performing bilinear sampling on these distanced dummies,

and then combining their results, the current pixel becomes informed of values found beyond its direct neighbors, enabling long-range decision-making. A trade-off to this is that, due to performing bilinear sampling on all n dummy locations, it ends up taking longer to complete our transformation. However, even if one does choose to include this method in their work, they should be mindful that the increased sampling range leads to more blurring in the resulting image. As such, this method should only be used when performing gradient descent, and not when creating the final transformed image.

2.3. Combating Geometric variation

When creating images, an individual object or a full scene will not always be captured the same way, resulting in differences in e.g. position, rotation, or even scale. For a machine learning model, this means that it needs to learn that even when an object is rotated or scaled, it is still the same object. This requirement increases the complexity of the task that the model is tackling, which in turn increases the size of the model and the amount of data that is required to sufficiently handle the task. Yet, in face of increasingly high demands, with higher resolution data and more complex datasets, we want to make sure that our models are as compact as possible, enabling their use on personal devices. To combat this blow-up, a commonly used approach is to separate the task of handling an object's pose from its actual classification, and turns it into a pre-processing step. With this, any image of a given object is transformed towards the same pose.

Existing methods that follow this logic, such as the Spatial Transformer Network and Deformable Convolutional Layer, are covered in this section, having achieved State Of The Art (SOTA) performance on datasets that contain geometric variation. It should be noted that while powerful, these methods still have their drawbacks. Notably, they do not generalize beyond the distribution that was found within the training dataset, preventing robustness against Out-of-Distribution data. Furthermore, even within distribution, learning to handle these transformations requires increased amounts of data. Our paper will focus on combating this issue, while providing results indicating its existence. As such, the methods that are covered here will serve as baselines that we will compare against.

2.3.1. Spatial Transformer Network

Based on the CNN architecture, the **Spatial Transformer Network** (STN) [8] is effectively an add-on that can be applied in front of any existing network. The STN architecture is shown in Figure 2.5. An example image or feature map showcasing the number '3' is passed to the STN's 'localization network', which directly predicts a transformation represented by θ. By applying the predicted transformation operation to the coordinate space of the image, and then performing the bilinear sampling operation discussed in section 2.2, we can obtain a transformed variant of our original input. By training the STN end-to-end alongside the downstream network, the STN is encouraged to reduce the variation seen across images in favor of improving downstream performance.

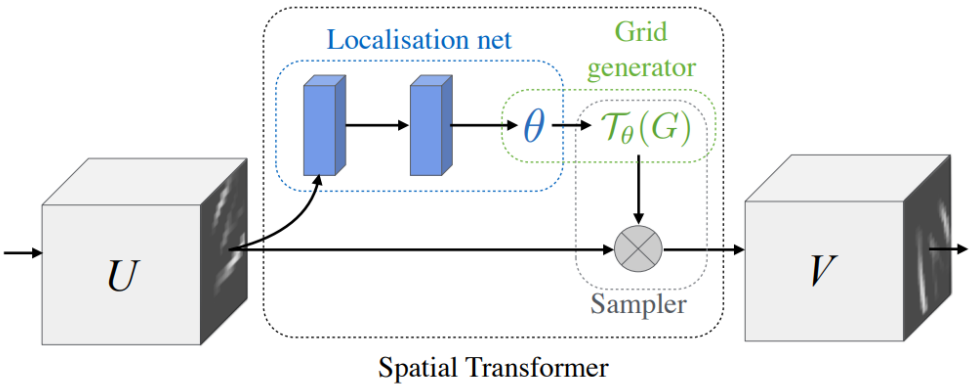

Figure 2.5: The Spatial Transformer Network architecture. Image from [8].

2.3.2. Deformable Convolutional Network

A notable line of successors to the Spatial Transformer Network focuses on incorporating its transformation capability directly into an existing architecture, bringing everything back into a single model. One such successor is the Deformable Convolutional Network (DCN), which incorporates transformations directly into a CNN layer. An example is showcased in Figure 2.6.

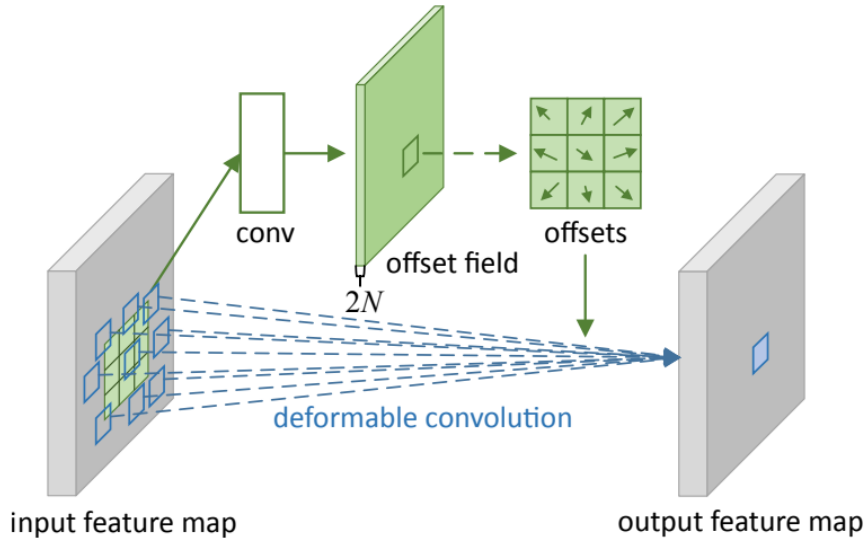

Figure 2.6: A Deformable Convolutional Network layer. By incorporating offsets into a kernel, our CNN increases its invariance to geometric variation. Image from [4].

Here, an individual DCN layer is shown. A normal CNN layer, as introduced in subsection 2.1.2 has a fixed kernel shape that is moved across the image, allowing for the derivation of information based on the values of directly neighboring pixels. While powerful, this means that the resulting feature map is affected by simple geometric variation, such as a slightly larger or rotated cat. The DCN changes this logic however and adds a 2D offset to each element of the CNN kernel. This offset is predicted by a regular CNN layer on the same feature map.

By predicting unique offsets for all elements in a kernel, the DCN is able to perform local transformations across all CNN layers. This is in contrast to the STN which applied a single global transformation to the entire image. As such, an object can have the same output even when scaled. With this, it is especially powerful in tasks such as image segmentation and object detection, which are local-based tasks [4].

2.3.3. Group Equivariant Convolutional Networks

A different approach to handling geometric variation focuses on making the model more **invariant** to certain transformations. Here, invariance means that the output of model computations remains identical for all objects belonging to a class, regardless of e.g. position, rotation, or scale. Notably, Group Equivariant Convolutional Networks [3] were proposed that handle this for rotations by creating rotated copies of CNN kernels. An example is shown in Figure 2.7, showing the result of 'p4' kernels.

If a CNN kernel focuses on horizontal edges, then one may expect it to behave differently when encountering an object flipped 90 degrees. This effect is visible here, where different rotations of the kernel result in the identification of either two disjoint parts, or a whole. By taking the same kernel, but rotated in 90 degree steps, we can ensure that at least one orientation of the kernel identifies the features that the model is looking for. By performing computations using these rotated kernels and then combining their results into one, it is possible to obtain similar outputs regardless of orientation.

2.3.4. Canonicalization

Building upon the idea of the GCNN, a field called canonicalization was brought about by Kaba et al. [10], which takes a slightly different approach. Namely, rather than combining the results of all feature maps directly, a score function is applied to each of the rotated feature maps, evaluating which orientation

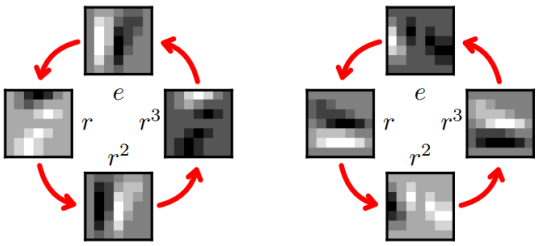

Figure 2.7: The result of a p4 kernel in a GCNN network. By rotating the kernel using 90 degree rotations we can ensure that the same features remain identified even under significant rotations. Image from [3]

brings the input sample closest to a **canonical** version. Here, the canonical version indicates a single pose or view that is shared for all samples belonging to a class. By performing this score function, downstream models can focus on reduced variation. A successor to this work, called FoCal [22], follows a similar philosophy to what we will be focusing on in our paper. Namely, it proposes to use **large foundation models**, which are large scale models pre-trained on substantial amounts of data, to serve as score functions for canonicalization. Having been trained on large amounts of real world images, it is possible to use these models to infer which pose is more natural, bringing the sample to its canonical variant. While effective across multiple scene conditions, the size of these models make them expensive to use in practice.

2.4. Image Congealing

The concept of reducing the variation seen across all images of a dataset is not new. This is the work of a field called Image Congealing, an example of which is shown in Figure 2.8. The concept was introduced by Miller et al. [19], which focuses on **group-wise alignment**, where images for a chosen class were aligned in favor of minimizing the joint entropy of the entire image set. Performed on digits with slight perturbations, this simple method allows for solid alignment for individual classes, reducing variation in favor of reduced complexity.

Figure 2.8: An example of Image Congealing performed on a singular class within the AffNIST dataset. Left: Samples of the digit '2' before image congealing. Right: Samples of the digit '2' after image congealing. Image from [1].

Building upon its success, subsequent work focused on improving the existing method. More efficient and effective loss functions were incorporated, such as MSE and MAE. Of these, the MAE loss function was found to be more robust to noise and occlusions, thanks to outliers having less impact as per subsection 2.1.3, enabling its use in real-world settings [1]. On the other hand, some methods focused on improving overall alignment performance by moving closer towards **pairwise alignment**, such as by incorporating a template-wise loss. Here, the current image of a class is compared directly to a chosen template representing that class. The inclusion of this loss ensured images would not drift away, thanks

to now having an anchor, and allowed for more accurate alignment [1].

However, while methods such as the STN have enabled strong downstream task performance on a variety of both single-class and multi-class datasets, the primary focus of congealing methods has been on aligning visually similar objects. With this, congealing methods have seen strong results in areas such as medical imaging, where scans of organs such as brains can be accurately aligned even in face of irregularities, enabling stronger performance on tasks such as segmentation [12, 7].

2.5. Data augmentation

While the previously mentioned methods are capable of reducing geometric variation, they are still susceptible to transformations that are out-of-distribution. Furthermore, when handling classes with limited representation, models may lack proper generalization capability. To increase the robustness of these methods for unexpected situations in real-world settings, data augmentation methods are utilized.

At its core, data augmentation focuses on the creation of artificial samples that fill gaps in/expand the training distribution. In the case of our problem focused on geometric variation, this can entail transforming data that is found in our training dataset by an unseen degree of rotation [16, 26]. If one wants a model that is fully robust, a train of thought may be to include any valid level of rotation, shear, scale, and rotation. If our model is large enough this can greatly improve its performance. However, this leads to a significant increase in the task's complexity, thereby further amplifying the problem that we are trying to solve.

2.6. Main architecture pre-requisites

Our paper is built upon a model that learns to reconstruct any image, similar to what was seen in the training set, in only a few steps. This model incorporates two techniques into an existing architecture. We will cover these techniques, as well as the architecture, in this section to facilitate understanding of how it works. We start off by introducing the techniques, as they build upon frameworks discussed so far. Namely, these are the ability to quickly adapt to new tasks, also known as Meta-Learning, as well as the ability to steer a model's output, known as Feature-wise Linear Modulations. After this, we introduce Implicit Neural Representations, the architecture that our model of choice is built upon.

2.6.1. Model-Agnostic Meta-Learning

Neural networks tend to focus on executing one particular task as accurately as possible through training. While the same model can be fine-tuned to perform well on a new task, this is typically inefficient. Meta-learning takes a different approach and asks: How can an individual model adapt quickly to new tasks, while using as few samples as possible? In 2017, the paper "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks" was introduced by Finn et al. [6]. While not the initial work, it has introduced the version of Meta-Learning that is used in the rest of this report.

In Model-Agnostic Meta-Learning, the training procedure is modified. A normal training loop focuses on the execution of a forward pass, a loss computation, and back-propagation, as described in subsection 2.1.1. This means that, for every batch of data, we perform back-propagation after executing this forward pass once. This leads to an updated set of model parameters, discarding the old set immediately. In meta-learning, we may refer to this procedure as the **inner-loop**. MAML takes a different approach however, where its intuition is shown in Figure 2.9.

Given a set of model parameters, aka its weights and biases, as well as n tasks T_i, we perform multiple inner-loop steps on a copy of our currently used parameters, where each update is performed by a different task. Once done, we perform a single backpropagation step through all of these consecutive inner-loop steps, updating our original set of model parameters. This full procedure is called an **outer-loop** step. By continuously performing these outer-loop steps, rather than teaching the model to perform accurately in one fixed task, we find a set of parameters that can adapt to new circumstances or tasks quickly in few training steps.

It should be noted that, while powerful, this method has one notable drawback: Memory efficiency. Every MAML update step is composed of multiple training steps, where every forward and backward pass has to be stored in memory until the MAML update step is complete. There are derivative works

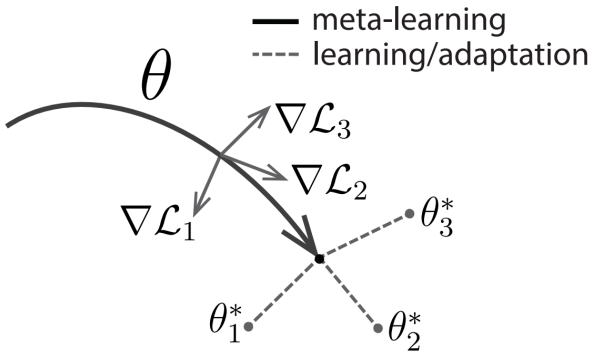

Figure 2.9: The MAML process, where the training process finds a set of parameters θ that allows it to quickly reach the optimal set of parameters θ_i^* corresponding to any task i . Here, $\mathcal{L}_i$ indicates the update direction corresponding to an individual task at the current iteration, which are all combined into a single update step. Image from [6].

that attempt to solve this issue, such as First-Order MAML and Reptile [20]. However, these methods either come at the cost of performance or enforce the assumption that one is updating a direct copy of the model's parameters during the inner-loop step. This becomes a concern when the inner-loop step is performed on a latent vector that influences our model, rather than the main model, as is the case in the architecture that this report is built upon.

2.6.2. Feature-wise Linear Modulations

The ability to steer a model based off an additional form of input that is independent of the model's normal input is a well sought after tool. Namely, by doing so, operations such as visual reasoning, the act of answering a question pertaining to a given image, become possible. A simple yet effective method of doing so was introduced by the paper "FiLM: Visual Reasoning with a General Conditioning Layer" by Perez et al. [21]. Here, FiLM stands for Feature-wise Linear Modulations, an example of which is shown in Figure 2.10.

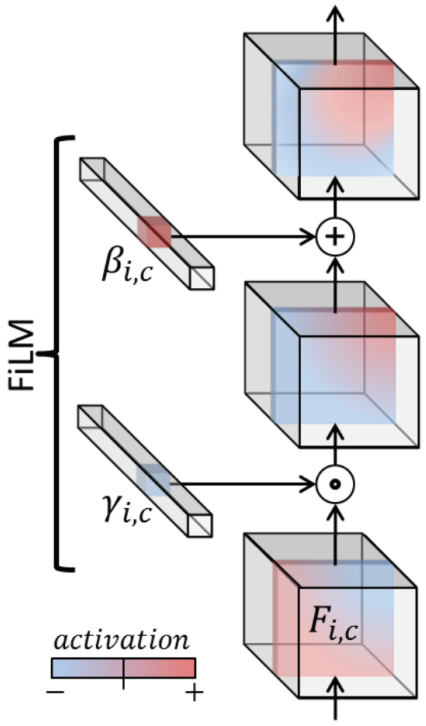

Figure 2.10: An overview of FiLM. Here, FiLM modifies the feature map $F_{i,c}$, where i is the current sample, and c is the current channel. This modification is performed through the use of a scalar, represented by $\gamma_{i,c}$, as well as a constant/bias term, represented by $\beta_{i,c}$. Image from [21].

As is shown, FiLM makes use of two vectors, namely $\gamma_{i,c}$ and $\beta_{i,c}$, where i indicates the current sample of a dataset and c indicates the current feature map. The manner in which these vectors are obtained does not matter, as long as they represent the additional input in some way. In the case of the original paper it is an encoded question pertaining to the current image. While the CNN focuses on deriving useful features from the original image, the two obtained vectors **modulate** the resulting feature maps through a scalar and bias term, in essence steering the model towards answering the latent question [21]:

$$FiLM(F_{i,c}|\gamma_{i,c}, \beta i, c) = \gamma_{i,c}F_{i,c} + \beta_{i,c} \tag{2.10}$$

2.7. Implicit Neural Representations

Implicit Neural Representations are an existing line of work focused on reconstructing data samples through regression. In the case of images, these models take as input the coordinates $x \in \mathbb{R}^{\not\vDash}$ of each pixel in an existing image and predict their corresponding color values $f(x) \in \mathbb{R}^{\not\Vdash}$ using a standard Neural Network. By doing so, data that can normally only be represented discretely, e.g. through a pixel grid, can now be represented continuously through this trained model, namely through the continuous input coordinates. This has a number of advantages: Due to its continuous nature, a single INR architecture can support any image resolution and inherently supports upscaling/downscaling. On top of this, it supports multiple data modalities, such as signals or 3D scenes, simply by changing the scope of the input [24].

Most famously, this type of model has been extensively researched in form of the Neural Representational Field (NeRF), capable of reconstructing 3D scenes. An example is shown in Figure 2.11. NeRFs are an extension of the INR that utilizes a 5D input, namely a 3D location (x, y, z) and a 2D viewing direction. Through this method, by using multiple reference images, the NeRF is not only able to reconstruct clear 3D scenes, but also allows the user to view it from any location and angle [18]. This has become a new standard for 3D scene representations in Computer Vision.

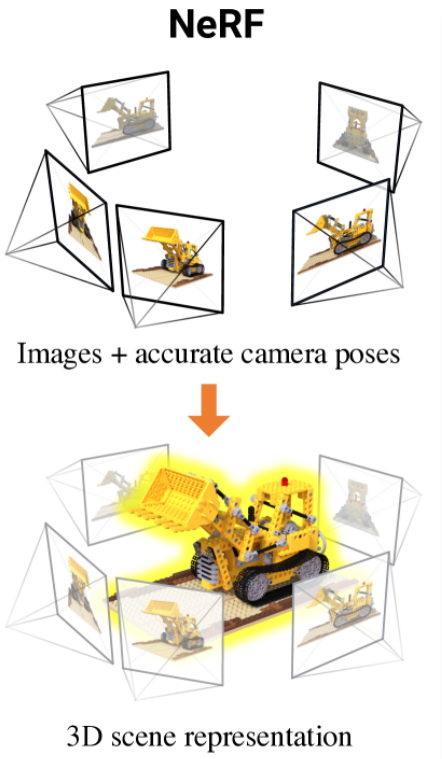

Figure 2.11: A Neural Representation Field (NeRF). Cropped image from [15].

Moving back to the world of images, INR models were improved greatly through the SiREN network [24], which utilizes sin activation functions, enabling high frequency details to be reconstructed.

2.8. From Data to Functa

The original INR framework reconstructs an individual image or 3D scene through multiple steps. If one wants to take advantage of how it represents data in a continuous format, such as for classification, this implementation becomes a bottleneck. Namely, a separate INR would have to be trained for each image. Even when taking a 'simple' dataset, such as the MNIST dataset, consisting of 60000 images of individual digits, this would result in similarly 60000 individual INRs. Each of these would have to be trained for 200+ steps, and their parameters, representing the continuous image, would have to be passed in full.

To address the above limitations and enable efficient classification using INRs, a powerful extension to the INR was published: "From data to functa" by Dupont et al. [5]. This work focuses on the idea of being able to reconstruct any image, part of a dataset, using a single global INR. Here, each image is represented by a "latent", which steers the reconstruction by making use of logic presented by FiLM layers. This way, the global model effectively becomes a **template**. Then, when given a new sample, only the latent is updated, which in turn modifies the output of our global INR. This allows it to reconstruct the desired sample, as is shown in Figure 2.12.

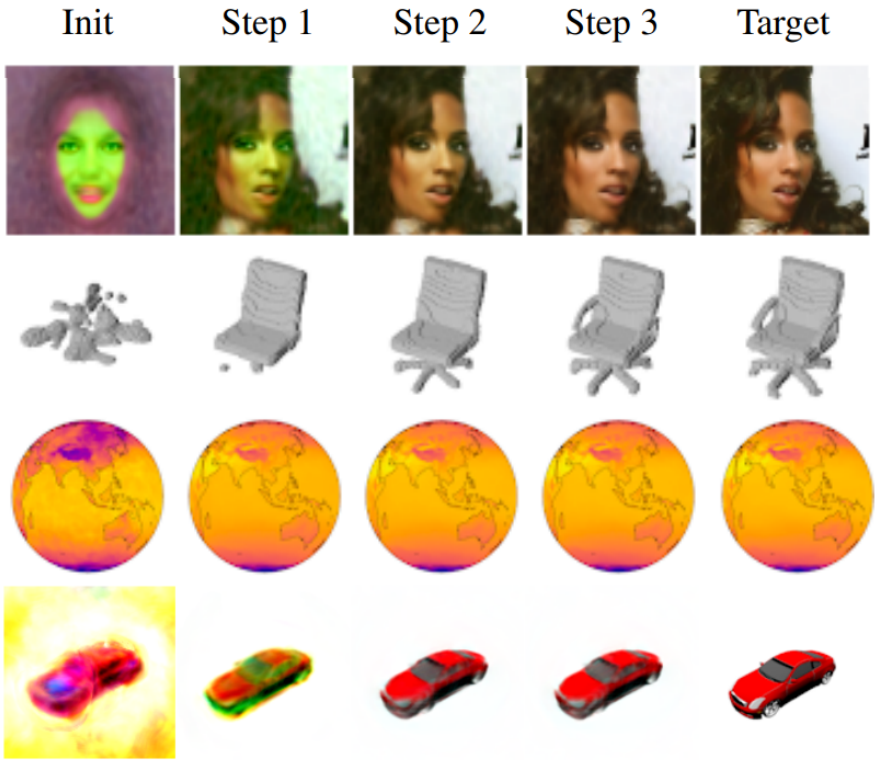

Figure 2.12: The Functa network in action on CelebA, an image dataset focused on images of celebrities, as well as three 3D datasets focused on shapes, globe data, and cars respectively. 'Init' showcases the learned global INR model without any modifications, showcasing a 'template' of each dataset, such as that of a human face. In only a few steps, it is able to move from this template to a full reconstruction of a sample found within the test dataset. Image from [5].

But how can we teach this global INR to find a template that is closest to all training images, when an INR normally takes multiple update steps to reconstruct the desired image? As is expected, the model cannot be trained to reconstruct each image in only one step. For this, we make use of the MAML framework discussed in subsection 2.6.1. To reiterate, the MAML framework allows a model to quickly adapt to a new task. In the case of our Data to Functa model, we need only tweak the wording, turning it into a framework that quickly adapts to a new image. More concretely, after updating only our image latents for 4 steps, we perform a single outer-loop step that updates our global model. This way, the INR model is still able to utilize multiple update steps to reconstruct a desired image, as before. However, as our meta-learning algorithm has found a template that is already close to each sample, the number of required update steps has been greatly reduced. For classification, we can then grab the latents,

trained to represent our images in a continuous format, and pass them directly into a classifier. While not SOTA in terms of classification performance, this results in a classifier model that is universal for any data modality [5].

Through this method, we now have an INR that represents a template of our dataset, with latents that can efficiently be trained to represent e.g. our images or 3D scenes as a 1D vector. We make great use of this structure in creating our method.

2.9. Additional terms

There are a few terms used throughout the paper that, while useful to know, do not affect the overall idea behind the paper. Nevertheless, some are described here for completeness.

se(2) parametrization is a different way of representing affine transformations, allowing gradients to be smoother. Under normal circumstances, an affine operation involving a rotation and translation is defined as follows:

$$A = \begin{bmatrix} cos(\theta) & -sin(\theta) & t_x \\ sin(\theta) & cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

This is the SE(2) variant of an affine transformation involving these operations. In turn, the mathematical lie group $se(2)$ is introduced, defined as follows, where ϕ represents the angle:

$$A = \begin{bmatrix} 0 & -\theta & t_x \\ \theta & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

By grabbing the matrix exponential of the $se(2)$ representation, the SE(2) variant can be obtained, coupling the translation and rotation while turning the rotation operation into a single scalar over the commonly used 6D affine. By using the SE(2) variant on the actual image transformation operation, and optimizing in the $se(2)$ space, gradients can be optimized on a mathematically smooth manifold. The SE(2) variant is used in the paper, which can be extended to the SE(3) variant for 3D settings, where it is even more powerful [2].

2.10. Datasets used

Controlled datasets are datasets that resemble real world data, but are heavily simplified to remove as many confounders as possible. For the experiments listed in this report six controlled datasets are utilized, which can be grouped into two categories, namely clean and augmented.

CLEAN

- MNIST [14]
- FashionMNIST [27]

AUGMENTED

- AffNIST
- FashionAffNIST
- SevereAffNIST [23]
- SevereFashionAffNIST [23]

The used variants of the MNIST dataset can be seen in Figure 2.13. The MNIST dataset, the original variant, consists of 70000 grey-scale samples of labeled digits from 0 through 9, of which 10000 samples are part of the test set. The goal is to classify these digits correctly based on the given image. While some slight variations in angle and translation can be seen, the original dataset is standardized heavily geometrically. The original AffNIST[1] dataset takes the original MNIST digits and adds transformations, adding geometric variation to the dataset. Namely, it uniformly modifies the digits through up to 20 degree rotations in either direction, 20% shears and scales in either direction, as well as translations to any valid position within the image bounds. Here, a valid position is any translation that results in no loss of information. More specifically, any pixel that is not fully black may not be pushed out of bounds. The presented Severe AffNIST variant scales this further, creating significant variations in digits. As

[1] https://www.cs.toronto.edu/~tijmen/affNIST/

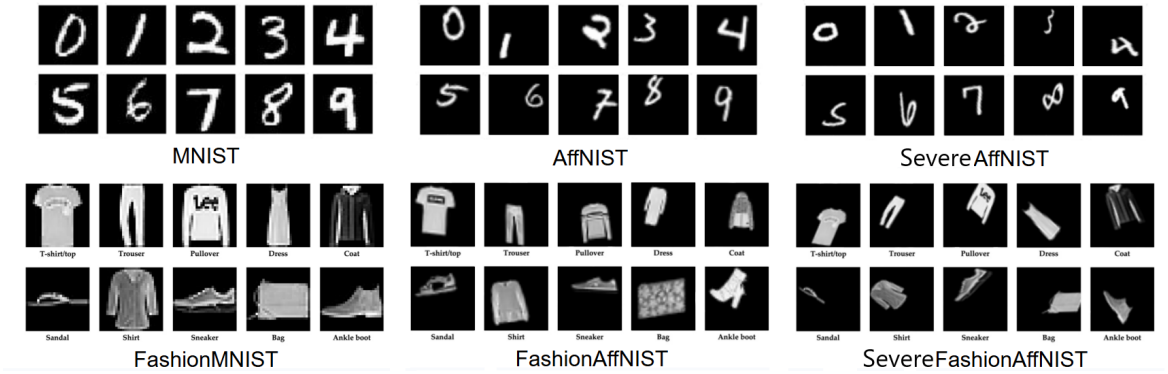

Figure 2.13: All variations of the MNIST datasets side by side: MNIST, AffNIST, SevereAffNIST - FashionMNIST, FashionAffNIST, SevereFashionAffNIST.

the original MNIST dataset is considered a simple problem to solve compared to real-world datasets, this serves as a bridge to real-world equivalent variations in geometry.

The FashionMNIST dataset is an existing counterpart to the MNIST dataset that replaces all numbers with clothes from Zalando article images, ranging from T-shirts and Trousers to Bags and Sneakers [27]. While the dataset contains even less variation in terms of rotation/translation, objects within classes can vary wildly, while also being more heavily textured. As such, this is considered a more difficult variation of the MNIST dataset, resulting in models obtaining worse classification performance. The augmented variants apply the same transformations as their MNIST counterparts dataset, providing a more challenging task.

References

[1] Roberto Annunziata, Christos Sagonas, and Jacques Cali. *Jointly Aligning Millions of Images with Deep Penalised Reconstruction Congealing*. 2019. arXiv: 1908.04130 [cs.CV]. URL: https://arxiv.org/abs/1908.04130.

[2] José Luis Blanco-Claraco. *A tutorial on $\mathbf{SE}(3)$ transformation parameterizations and on-manifold optimization*. 2022. arXiv: 2103.15980 [cs.RO]. URL: https://arxiv.org/abs/2103.15980.

[3] Taco S. Cohen and Max Welling. *Group Equivariant Convolutional Networks*. 2016. arXiv: 1602.07576 [cs.LG]. URL: https://arxiv.org/abs/1602.07576.

[4] Jifeng Dai et al. *Deformable Convolutional Networks*. 2017. arXiv: 1703.06211 [cs.CV]. URL: https://arxiv.org/abs/1703.06211.

[5] Emilien Dupont et al. *From data to functa: Your data point is a function and you can treat it like one*. 2022. arXiv: 2201.12204 [cs.LG]. URL: https://arxiv.org/abs/2201.12204.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. 2017. arXiv: 1703.03400 [cs.LG]. URL: https://arxiv.org/abs/1703.03400.

[7] Juan Eugenio Iglesias et al. "Joint registration and synthesis using a probabilistic model for alignment of MRI and histological sections". In: *Medical Image Analysis* 50 (2018), pp. 127–144. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2018.09.002. URL: https://www.sciencedirect.com/science/article/pii/S1361841518306972.

[8] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Advances in neural information processing systems* 28 (2015).

[9] Wei Jiang et al. *Linearized Multi-Sampling for Differentiable Image Transformation*. 2019. arXiv: 1901.07124 [cs.CV]. URL: https://arxiv.org/abs/1901.07124.

[10] Sékou-Oumar Kaba et al. *Equivariance with Learned Canonicalization Functions*. 2023. arXiv: 2211.06489 [cs.LG]. URL: https://arxiv.org/abs/2211.06489.

[11] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[12] Nefeli Lamprinou, Nikolaos Nikolikos, and Emmanouil Z. Psarakis. "Groupwise Image Alignment via Self Quotient Images". In: *Sensors* 20.8 (2020). ISSN: 1424-8220. DOI: 10.3390/s20082325. URL: https://www.mdpi.com/1424-8220/20/8/2325.

[13] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[14] Yann Lecun, Patrick Haffner, and Leon Bottou. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86 (Dec. 1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[15] Chen-Hsuan Lin et al. *BARF: Bundle-Adjusting Neural Radiance Fields*. 2021. arXiv: 2104.06405 [cs.CV]. URL: https://arxiv.org/abs/2104.06405.

[16] Francisco López de la Rosa et al. "Geometric transformation-based data augmentation on defect classification of segmented images of semiconductor materials using a ResNet50 convolutional neural network". In: *Expert Systems with Applications* 206 (2022), p. 117731. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.117731. URL: https://www.sciencedirect.com/science/article/pii/S0957417422010120.

[17] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". en. In: *Bulletin of Mathematical Biophysics* 5.4 (Dec. 1943), pp. 115–133.

[18] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV]. URL: https://arxiv.org/abs/2003.08934.

[19] Erik G. Miller, Nicholas E. Matsakis, and Paul A. Viola. "Learning from one example through shared densities on transforms". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)* 1 (2000), 464–471 vol.1. URL: https://api.semanticscholar.org/CorpusID:2699786.

[20] Alex Nichol, Joshua Achiam, and John Schulman. *On First-Order Meta-Learning Algorithms*. 2018. arXiv: 1803.02999 [cs.LG]. URL: https://arxiv.org/abs/1803.02999.

[21] Ethan Perez et al. "Film: Visual reasoning with a general conditioning layer". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[22] Utkarsh Singhal et al. *Test-Time Canonicalization by Foundation Models for Robust Perception*. 2025. arXiv: 2507.10375 [cs.CV]. URL: https://arxiv.org/abs/2507.10375.

[23] Soumen Sinha, Jan van Gemert, and Alexander Gielisse. *Are accidental scene conditions learnable?* 2026. URL: NOTYETAVAILABLE.

[24] Vincent Sitzmann et al. *Implicit Neural Representations with Periodic Activation Functions*. 2020. arXiv: 2006.09661 [cs.CV]. URL: https://arxiv.org/abs/2006.09661.

[25] Charles Thiery. *Understanding and Visualizing Decision Boundaries in a One-Hidden-Layer Neural Networks*. Jan. 2025. URL: https://medium.com/@Charles_Thiery/how-artificial-neural-networks-use-non-linear-decision-boundaries-for-data-classification-9960d80916cf.

[26] Zaitian Wang et al. *A Comprehensive Survey on Data Augmentation*. 2025. arXiv: 2405.09591 [cs.LG]. URL: https://arxiv.org/abs/2405.09591.

[27] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. arXiv: 1708.07747 [cs.LG]. URL: https://arxiv.org/abs/1708.07747.

3

Scientific Paper

AxFuncta - Fit instead of Predict for Accidental Scene Conditions.

Vladimir Rullens, Alexander Gielisse, Jan van Gemert
Computer Vision Lab
Delft University of Technology
v.c.j.r.rullens@student.tudelft.nl

Abstract

In recent years, strong progress has been made in creating learnable affine-equivariant models for downstream tasks such as classification. However, these models encounter increased data requirements to represent all possible transformations due to greater task complexity, while having been shown to generalize poorly to out-of-distribution data. In this work, we introduce a test- time approach for generalizing to out-of-distribution data. Namely, by utilizing a network trained to reconstruct any image that is part of a standardized training distribution, our model can infer an affine transform that moves new samples in- distribution by minimizing their reconstruction loss. With this, this approach closely matches the work of Spatial Transformer Networks, which instead learn to transform data, and inverted neural renderers for pose estimation. Through experiments, we show that this method contains a strong level of out-of-distribution translation and scale invariance, as well as a small level of rotation invariance. Namely, we show that it can handle significant transformations beyond those produced by commonly used benchmarks such as AffNIST. Using this strength we show that this method excels especially in low data regimes, outperforming existing competitors.

1. Introduction

Computer vision models are susceptible to geometric transformations frequently seen in real-world settings, such as cats that lay on their sides or doors viewed from angles. This results in increased task complexity due to the need to 'learn' to handle the geometric transformations found in the training data. Even worse, it has been shown that if the transformations were outside the scope of what was seen while training, then the model may be unable to produce a correct classification [37]. As such, a desirable property of models is to be either invariant or equivariant to these transformations, as to enable smaller and more robust models.

Strong progress has been made in combating this issue through the introduction of different strategies. Most notably, methods such as the Spatial Transformer Network were introduced [15], which learn to remove geometric variation in images before sending them to a model of choice, improving their performance. However, despite being trained to 'correct' images, it has been shown that they are still unable to handle out-of-distribution transformations [37]. This leads to problems as shown in Figure 1, where the main object in an image is misclassified when ro-

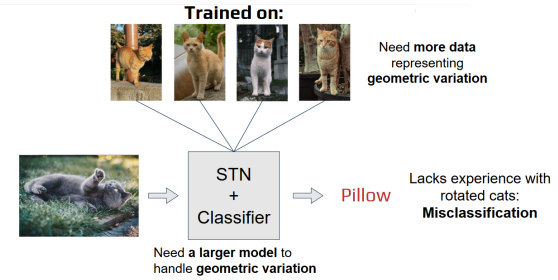

Figure 1. Problem example: Despite using models such as the STN, an architecture trained to remove geometric variation such as rotations, images will be misclassified if the subject's pose is not represented by the training dataset. As a result we need more data and larger models to handle this data for performance.

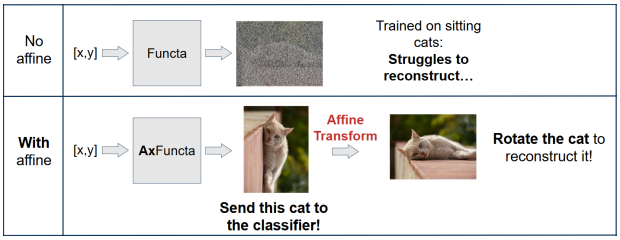

Figure 2. AxFuncta concept: A model that has seen a limited set of poses, e.g. only sitting cats, may be unable to reconstruct unseen poses, e.g. cats laying down. By giving the model the freedom to apply a transformation to its reconstruction, unseen poses can be fit back towards the training distribution. This allows the potential for a downstream model to work with any pose, while keeping its requirements minimal.

tated, despite being trained to take transformations into account. In turn, data augmentation methods were proposed that increase the size of the training distribution through the creation of artificial samples, obtained from transforming the original training data [5, 45]. While this improves model robustness, it requires handling more data and therefore larger models, while being bound to the size of the introduced transformations, leaving them unreliable to handle unseen, out-of-distribution data.

In this work, we propose a 'fit' instead of 'predict' approach for handling out-of-distribution (OOD) data, enabling a level of robustness to OOD translation, scaling, and rotations, without training a model to do so. We show this in Figure 2. Namely, we utilize a FUNCTA network, which is a model that we train to reconstruct a limited distribution of known poses in few steps. When presented with a new sample that is found outside of this distribution, the network is expected to create a poor reconstruction, as it has not learned to handle the sample's unseen pose. We take advantage of this by giving the network the ability to transform these samples while reconstructing. This enables it to move samples towards a pose that the network is familiar with, such that it can accurately reconstruct them as per its original task. This is reminiscent of the behavior of learning-based methods, which improve poses in pursuit of accuracy. The resulting transformed sample can then be passed to a downstream model, such as a classifier, largely removing the previously discussed bottleneck caused by geometric variation. As a result, our downstream models can stay compact, reducing data requirements. By building on top of the FUNCTA, we leave our method open to generalize to other data modalities such as 3D scenes or signals, enabling a potential universal OOD generalization method.

Our contributions are as follows:

- We propose AxFuncta, a novel 'fit' instead of 'predict' approach for OOD generalization, built on top of recent work by Dupont et al. [10], applicable to either the input or output of the adopted architecture.
- Through experiments, we show strong OOD generalization on translation and scale, as well as a level of OOD generalization on rotations. In turn this enables our method to significantly outperform our competitors in OOD settings.
- We also show improved performance in low-data regimes when handling in-distribution data, bringing about a strong method in situations where data is scarce.
- Finally, an ablation study is presented showcasing the necessity of e.g. latent regularization, padding, and an improved sampling approach by Jiang et al. [16] to achieve best results.

2. Related Work

Learning-Based Methods The main line of work focused on the reduction of geometric variation in datasets utilizes deep-learning models. Spatial Transformer Networks (STNs) [15] attach a localization network in front of an existing model. This network predicts a transformation and applies it to the original input before handing it to the downstream model, enabling improved downstream performance by learning to canonicalize samples. Subsequent work has split into multiple branches, such as improvements to the original STN through compositional iterative [1, 22] and probabilistic [33, 34] counterparts, as well as modules built into CNN filters [9] and transformer attention layers [40]. However, learning-based approaches such as the STN have been shown to lose accuracy when handling data with increased geometric variation, cannot generalize transformations to other classes, and struggle out-of-distribution [12, 37].

Joint Image Set Alignment. To remove geometric variation between images in a set 'congealing' methods were introduced, which align all images part of a set in an unsupervised manner. Doing so has been shown to have use cases in areas such as medical imaging [14] and image editing [31]. Initially, this was done solely in single-class settings using a sum of entropy loss across pixels [19]. However, due to this being computationally inefficient, a least-squares error was quickly introduced as an effective alternative [8], with the option of taking the mean absolute error being shown to be more robust against distortions such as occlusions [3].

In more recent years, GANgealing [31] pushed this approach towards large-scale real-world datasets by utilizing a pre-trained GAN to generate artificial samples, effectively adding data augmentation to congealing for improved performance. Neural Congealing [30] on the other hand extended this to real-world datasets by aligning features that are derived by an existing large foundation model. However, these congealing methods are built around the alignment of visually similar objects and as such lack multi-class capabilities on their own, making them unsuitable for tasks such as classification.

Methods that jointly align and cluster image sets have been introduced [24, 29, 43], yet these are underexplored in terms of their ability to aid directly in tasks such as classification, and tend to incorporate STNs, bringing us back to the presented problem. Nevertheless they present a powerful paradigm, leading to a level of adoption in our work.

Canonicalization A recent approach towards equivariant models is Canonicalization, which has been explored in areas such as image classification [17, 35, 39], 3D scenes [21, 36], and graph data [26]. This approach is simi-

lar to that of methods such as the STN, where an invertible transformation is found that moves a data point to its canonical representation, on which any existing model can then operate. Here however, the "removal" of existing transformations is handled by a set-based canonicalization function $h(\mathrm{x})$. Given some input x, Kaba et al. [17] defines the following optimization approach:

$$h(\mathrm{x}) \in \underset{\rho(g)\in\rho(G)}{\arg\min}\ E\left(\rho(g)^{-1}\,\mathrm{x}\right) \tag{1}$$

Here, $\rho(g)$ represents the applied transformation part of a set of possible transformations $\rho(G)$, while the constrained energy function E represents the distance to the canonical representation. In the case of Kaba et al. they make use of a learning based approach that includes a Group Equivariant Convolutional Network [7] (GCNN), where the score function is applied to its rotated kernels to determine the correct angle. Subsequent work has shown the notable effectiveness of canonicalization on pre-trained models by using probabilistic canonicalization priors that match the downstream model's expected rotation [28]. Recently, FoCal [36] extended capabilities to scene conditions, such as lighting and contrast, by using learned visual priors from two existing pre-trained large foundation models.

We take a similar approach to the latter and focus on a 'fit' instead of 'predict' approach, but make use of a Functa network trained to reconstruct only known poses. This enables the use of our approach on datasets unsupported by these models, while removing the need for large foundation models.

Implicit Neural Representation (INR). Implicit Neural Representations (INRs) are models that learn to reconstruct individual data samples, such as images [38, 42], or 3D scenes [23, 27]. In the case of images, this is performed by taking the $(x, y) \in [-1, 1]$ coordinates of a pixel in image space and predicting its corresponding $(r, g, b) \in [0, 1]$ color values. This allows for the creation of a continuous variant of our original sample, while being agnostic to its original resolution. SiREN [38] improved this idea using sinusoidal representation networks, allowing for the reconstruction of fine details. In the meantime, Neural Radiance Fields (NeRF) brought about an extension towards 3D scenes, bringing about many follow-up works [23, 27, 44]. However, this formulation only allows the reconstruction of an individual data instance per INR, preventing its usage in downstream deep learning tasks.

To allow for the reconstruction of any image within a dataset, generalizable INRs were introduced [6, 10, 18]. One such work, From Data to Functa [10], introduces the Functa network. This network incorporates a latent vector, specific to each sample, that modulates a fixed INR through the use of Feature-wise Linear Modulations [32]. To train the INR itself, meta-learning is used to find a representation of a training set that is only a few steps away from reconstructing its samples [11]. Subsequent work includes a focus on improving its classification performance through a modified latent representation [4, 25], with recent work extending this to high-resolution datasets while utilizing an end-to-end training approach [13]. In a similar vein to image congealing, we utilize a Functa's modulated representation as a variable 'template' to move out-of-distribution samples towards. To the best of our knowledge, we are the first paper to utilize generalizable INRs in this fashion.

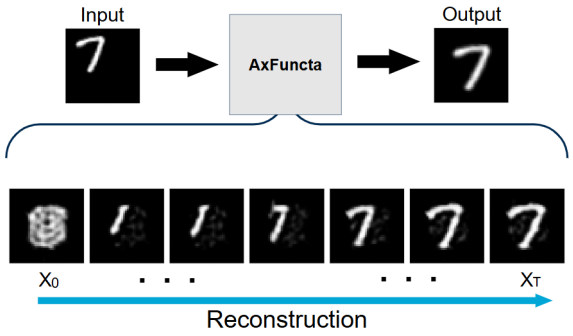

Figure 3. The AxFuncta method in action. By optimizing for reconstruction quality while allowing for transformations to take place, we can remove geometric variation from a new sample using INRs. At t=0 the initial global template can be seen. Images shown are the INR's attempted reconstruction before the affine is applied.

3. AxFuncta

Our approach builds upon the Functa network described in section 2, and is shown in Figure 3. At train-time the behavior of this method is left as-is, enabling it to learn how to reconstruct images from our training set in only $t = 4$ gradient steps. In turn, the model finds a 'template' of our dataset, which can modify itself freely to fit a newly presented sample.

Instead, our method modifies the test-time behavior of a trained Functa model f_θ. When presented with an out-of-distribution sample, a trained generalizable INR may struggle to reconstruct it accurately, reminiscent of problems found with learning-based methods. Here however, we take this problem and turn it into an advantage by jointly optimizing a learnable affine transformation during the reconstruction process, represented by the vector A. This enables the model to move the sample back in distribution, where it can reconstruct it more accurately and thereby minimize its reconstruction loss. An example of this is shown in Figure 3, where the reconstruction quality of a number '7' increases as it approaches its canonical pose. This results in the following network, where $I \in \mathbb{R}^C$ represents the image

for which the color values are reconstructed at input coordinates $x \in [-1, 1]^2$. Following the approach in Data to Functa [10], $z \in \mathbb{R}^k$ represents the latent of size k, which modulates our network:

$$\hat{I}(x) = \text{warp}(f_\theta(x, z), A) \tag{2}$$

Here, A is a 4-dimensional latent containing elements of the Euclidean Group $se(2)$, as well as a scale parameter. After converting the latent into an affine matrix we can use it to apply a transformation directly to the reconstructed image using $\text{warp}(I, A)$, as is shown, or to the set of coordinates x that is passed as an input, moving the coordinate space itself. Through this procedure, the model can freely reposition its 'reconstruction' such that it aligns with the geometric orientation of our target sample, in an effort to minimize its reconstruction loss. Then, once A has converged, we can apply the inverse of our resulting transformation on our initial sample to obtain a canonicalized variant that matches our training distribution, without any remaining reconstruction artifacts.

While the original Functa network can reconstruct images or 3D objects in only a few gradient steps t, more steps are required to converge to an optimum for the affine vector A. As such, we are required to increase the number of steps t to find this optimum. However, rather than increasing it at train time, we instead opt to increase it only at test-time. This ensures that the Functa's reconstruction speed benefit remains, allowing it to quickly adapt to changes in A. Then, to constrain the latent from overfitting to our unseen sample, we apply L2 regularization to said latent at test-time. This results in the following inner-loop loss function L_{test}, where N represents the number of samples in a batch:

$$L_{test} = \frac{1}{N} \sum_{i=0}^{N} (\text{warp}(f_\theta(x, z), A) - I_i)^2 + z_i^2 \tag{3}$$

To further improve gradient flow during our fit operation, we make use of the linearized sampler proposed by Jiang et al. [16]. , which artificially increases the range in which the bilinear sampling operation is performed during an image transformation. As a result, gradients flow beyond a pixel's direct neighbors, enabling an improved fit.

3.1. Training set standardization.

The currently proposed method works under the assumption that the training dataset is pre-aligned, such as is the case with the MNIST dataset. If data is not yet aligned, this may lead to our model incorrectly transforming new samples due to its overwhelming generalization capability. Namely, rather than learning to reconstruct specific poses, this results in the Functa learning to adapt to a wide range of poses. This is an issue, as real-world datasets contain significant levels of geometric variation. An example is shown in Figure 4, where we train our logic presented so far on the AffNIST dataset. As is visible, samples do not properly align to individual datasets, instead opting to purely maximize scale.

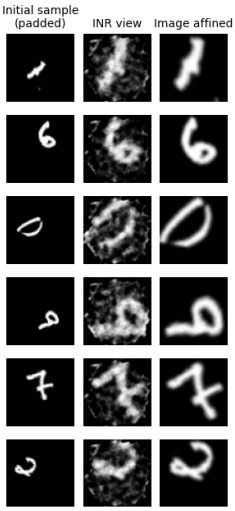

Figure 4. Showcasing AxFuncta's results when performed on an unstandardized training dataset. Here, AxFuncta is trained on the raw AffNIST dataset, which has geometric variation built-in. Left column: The original image. Middle column: Functa's final reconstruction w/ regularized latent. Right column: The final transformed image.

To enforce proper alignment, we therefore perform a method similar to existing work on image congealing by Annunziata et al. [2]. Similarly to our approach, it standardizes individual, rather than all, classes in datasets by optimizing reconstruction performance and alignment to a chosen template. In the meantime latent capacity is limited through an adaptive regularization function, which enforces the dataset to align in favor of a simpler latent.

Here, we first train our generalizable INR for 4 epochs, providing a warm-start that enables it to find initial reconstructions that it can work from. For subsequent epochs, we enable iterative refinement of our dataset, reminiscent of both congealing and pose estimation work. Namely, after the initial reconstruction phase, we update our estimated 'pose' for each sample every epoch. We then utilize the following loss function L_{train}, incorporating the latent loss from the previously mentioned congealing method, as well as an MSE class-wise alignment loss based on early congealing work:

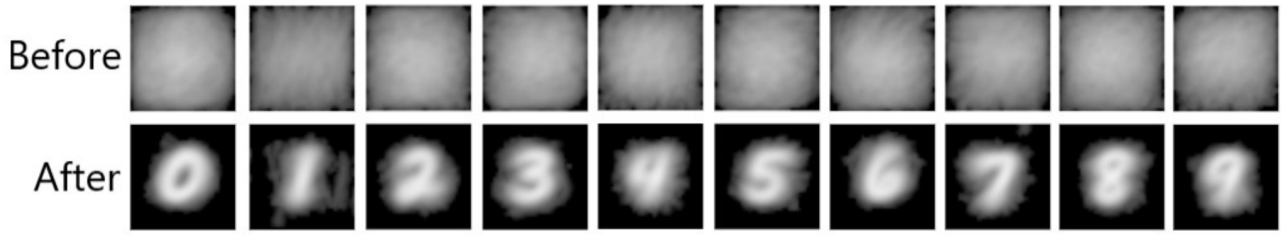

Figure 5. A before and after view of our train-time standardization process displayed using the means of clustered classes. Results showcase that a template of our digits comes into view, indicating the effectiveness of the alignment procedure. Performed on a subset of the 'AffNIST' dataset (5k samples).

$$L_{train} = \frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{|C(i)|}(\hat{I}(x_j) - \hat{I}(x_i))^2 + \frac{1}{N}\sum_{i=0}^{N}(\hat{I}(x_i) - I_i)^2 + g(|z_i|) \quad (4)$$

Here, $|C(i)|$ indicates the number of samples that are of the same class as sample i, handling our class-wise alignment loss. Following Annunziata et al.'s implementation $g(z_i) = w^T z_i$, where z is our latent enforced to only contain positive values, while w is a weighted vector $w = [w_1, ..., w_k]^T$. In this weighted vector, each element $w_l = l / \sum_{l=1}^{k} l$. Examples of a before and after using this procedure are shown in Figure 5. In the 'Before' row, we display the mean of all samples belonging to each class on the AffNIST dataset, which is a dataset containing transformed digits. 'After' showcases the level of standardization that our train-time procedure is able to achieve. While some digits, such as some samples of the number '1' are separated, it is evident that digits are aligned well in terms of translation.

4. Experiments

The specifics of our models are described in Appendix A. All code is implemented in Pytorch 2.6.0 and executed on a single NVIDIA A40 GPU using seeds 42 and above for the respective iterations. We opt to tune using training step iterations instead of epochs to streamline the process across different dataset sizes while allowing for fairness. The batch size is 48 across the board to align with the Functa model's training requirements.

Datasets. Experiments are performed on the MNIST [20] and FashionMNIST [41] datasets. To produce geometric variation, counterparts to the above datasets are made based on the AffNIST dataset[1], termed AffNIST and FashionAffNIST respectively. These uniformly transform their counterparts using scale and shear operations within the range of 20%, rotations up to 20 degrees in either direction, and unbounded translations, after a padding of 6 in either direction. To ensure parity between MNIST and AffNIST datasets in our experiments, we pad the MNIST variants similarly. We also make use of the severe variants of these datasets which we will here call SevereAffNIST and SevereFashionAffNIST, following the same specifications as used in [37]. These severe variants take the same logic used by the AffNIST dataset, but scale the severities of the operations, each containing 60x60 images.

Baselines. We compare our method against existing learning-based methods, namely a base CNN, STN[2] [15], deformable CNN [9], as well as the learnable canonicalizer by Kaba et al. [17] implemented using their EquiAdapt package [3]. The CNN backbone is described in Appendix A, which is universal for all methods to ensure that only the methods themselves influence the outcome. We also compare our method against invariant models such as the p8 variant of the Group Equivariant Convolutional Network [7].

Reliability notice. We note that unexplained behavior, potentially due to a bug, was found nearing the deadline of our research where the fit procedure of AxFuncta did not make use of rotations for part of our train-time results in Experiment 3. This is notable in our train-time standardization qualitative assessment and should be kept in mind when viewing these results, as it means that they do not indicate the effectiveness of performing rotations. **OOD results are unaffected** and as such are our primary focus for our ablations, showcasing the effectiveness of our fit procedure.

4.1. Experiment 1: OOD generalization

Before evaluating the effectiveness of our method on our initial hypothesis, we test whether our method works through an OOD generalization test, validating its effectiveness on standardized training datasets. To perform this

[1] https://www.cs.toronto.edu/~tijmen/affNIST/

[2] https://docs.pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html

[3] https://github.com/arnab39/equiadapt

Model	MNIST ->AffNIST	FashionMNIST ->FashonAffNIST
CNN	20.09%	26.01%
STN	21.79%	29.00%
DeformableCNN	24.31%	29.68%
GCNN	22.52%	26.00%
EquiAdapt	19.73%	23.32%
AxFuncta (image space)	**83.85%**	**57.05%**
AxFuncta (coordinate space)	50.67%	32.97%

Table 1. OOD accuracy on the 'clean ->augmented' setting across 2 methods and all models. Here, all methods are trained on the clean dataset and evaluated on the augmented dataset, simulating an OOD environment. We evaluate our method, AxFuncta, on two variants. These variants include image space, where we apply an affine transformation to the reconstructed image, as well as coordinate space, where the same transformation is instead applied to the input coordinates. Results indicate that the image space variant outperforms all baselines, as well as the coordinate space variant, by a substantial margin. Average across 5 runs.

experiment, we train all methods on the clean-only datasets, MNIST and FashionMNIST, while evaluating them in terms of % accuracy on the transformed datasets, AffNIST and FashionAffNIST respectively. Our method, AxFuncta, is performed without standardizing our training set. This is possible as the MNIST and FashionMNIST datasets are pre-standardized. As noted in section 3, there are multiple potential targets for our affine latent A. As such, we perform our experiments on two variants of our proposed method AxFuncta, applying transformations to the reconstructed image (image space) or the input coordinate space (coordinate space). The results are showcased in Table 1.

As is shown in our results, despite all methods being given an equivalent CNN, Meanwhile, our method is able to obtain significantly better OOD performance through a fit instead of predict approach. While the OOD performance in itself is a noteworthy success, this validates that our method shows promise under the right circumstances, by bringing unseen digits to a standardized distribution despite having never been trained to do so. We note that this does come at a cost to computation: The existing method performs this across the full test dataset across 58 minutes, whereas competitors can do the same computation in a matter of seconds.

Furthermore, our results improve when making the target of our 'fit' approach the image space over the coordinate space, resulting in up to 33.18% higher accuracy. Our hypothesis is that this is due to a weakened gradient flow to the input coordinates, worsening the quality of our transformation. Focusing on beating our baselines in terms of performance, subsequent experiments incorporate the image space variant of AxFuncta. Qualitative examples of our fitting approach are shown in Appendix B.

4.2. Experiment 2: Increasing perturbation strength

To establish which perturbation types our method can handle best, we evaluate the OOD strength of our method compared to existing methods on three supported geometric perturbation types:

- Translation: [-0.4, 0.4] (using a [-1,1] bound coordinate space)
- Rotation: [-90, 90] degrees
- Scale: [0.5, 2.0]

Each perturbation type is executed individually across a range of perturbation strengths [0.0, 0.2, 0.4, 0.6, 0.8, 1.0], which serve as multipliers to the above values. We perform these experiments on the FashionMNIST dataset, with perturbations being applied solely to the test dataset, resulting in a consistent train distribution across all data points. As per experiment 1, all methods are evaluated in terms of % accuracy. Results are showcased in Figure 6.

As is shown in the figure, existing methods notice significant drops in accuracy as perturbation strength increases, with there being seemingly no benefit between most models. The exception to this is EquiAdapt, which outperforms its competitors on rotations. An additional test to investigate this behavior can be found in Appendix D, giving a possible explanation. Meanwhile, the accuracy of our method, AxFuncta, is much more stable when it comes down to Translation and Scale, showcasing a strong level of invariance. For rotations on the other hand, we see that our method underperforms compared to existing methods, likely as a result of encountering local minima during the fitting process. In turn, our rotations only manage to tie all but EquiAdapt as the perturbation strength increases, indicating a weakpoint.

To further understand these results, qualitative results are showcased in Figure 7. Here, for the digit '2' in the AffNIST dataset, we showcase the mean of all test samples, with 'Before' showing the initial perturbed test datasets that all models were evaluated on, while 'After' displays the result obtained from applying our AxFuncta method.

When increasing the perturbation strength, we see that the mean sample in our test set shifts from a distinct '2' to a blurry landscape. As is expected, this results in a reduction in accuracy for our baselines due to digits no longer being within the expected distribution. However, AxFuncta

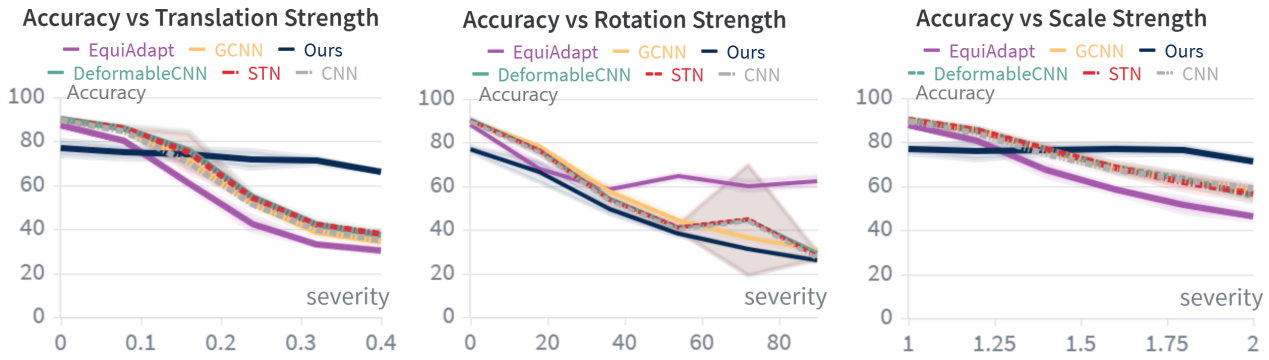

Figure 6. Accuracy of all methods on the FashionMNIST dataset when applying increasingly large perturbations. Our method outperforms existing baselines on OOD translation and scale perturbations, but underperforms on rotations.

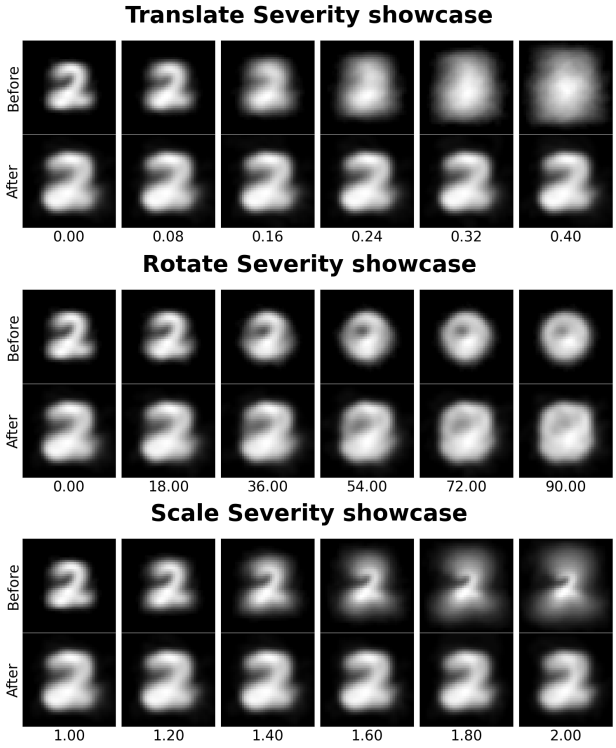

Figure 7. Mean perturbed images across an increasing perturbation strength for our tested perturbation types: Translate (top), Rotate (middle), Scale (bottom). 'Before' showcases the test set that each method is evaluated on, while 'After' showcases the result of AxFuncta's fit procedure. These results showcase that AxFuncta is able fix translation and scale perturbations with no noticeable drawbacks across increasing severities. However, there is a slight increase in scale overall that can affect performance. Rotations on the other hand see some improvements, but remain a weakness. These results line up with and explain our findings in Figure 6

.

manages to recover the initial distribution almost perfectly for both translate and scale perturbations, even at the highest setting. This in turn explains the near horizontal accuracy line that was also seen in our quantitative graphs. The scale of our recovered mean digits is a bit larger than our initial distribution however, which explains the slight reduction in accuracy even at no perturbation strength. On the other hand, for rotations, the lack in ability to perfectly rotate any digit is clearly visible. Nonetheless, even for rotations a template of a '2' sticks around longer when our AxFuncta method is applied, with it still being somewhat visible even at 54 degree rotations, which cannot be said for our initial perturbed test set. This explains how its performance manages to catch up to our baselines despite its initial disadvantage.

For a side-by-side view of individual samples, we direct the reader to Appendix B.

4.3. Experiment 3: Varying datasize

As has been established in section 3, and as has been shown in the previous experiments, our method works under the assumption that the training set is standardized prior to training our Functa network. Having seen the extent of this, we move on to our main hypothesis: A 'fit' instead of predict approach results in lower data requirements. As such, the subsequent experiments are performed with the full approach in mind, namely through the inclusion of train-time training set standardization.

To evaluate the effectiveness of our method on smaller datasets, we trained all models on varying dataset sizes. Our baselines were executed on both the clean-only and transformed-only datasets, while our method was only evaluated on the transformed setting. Here, the clean-only results serve as an upper-bound, indicating the maximum possible accuracy when a model can mostly ignore appearance-specific information, while transformed-only indicates the expected lower-bound. The goal of this experiment is to

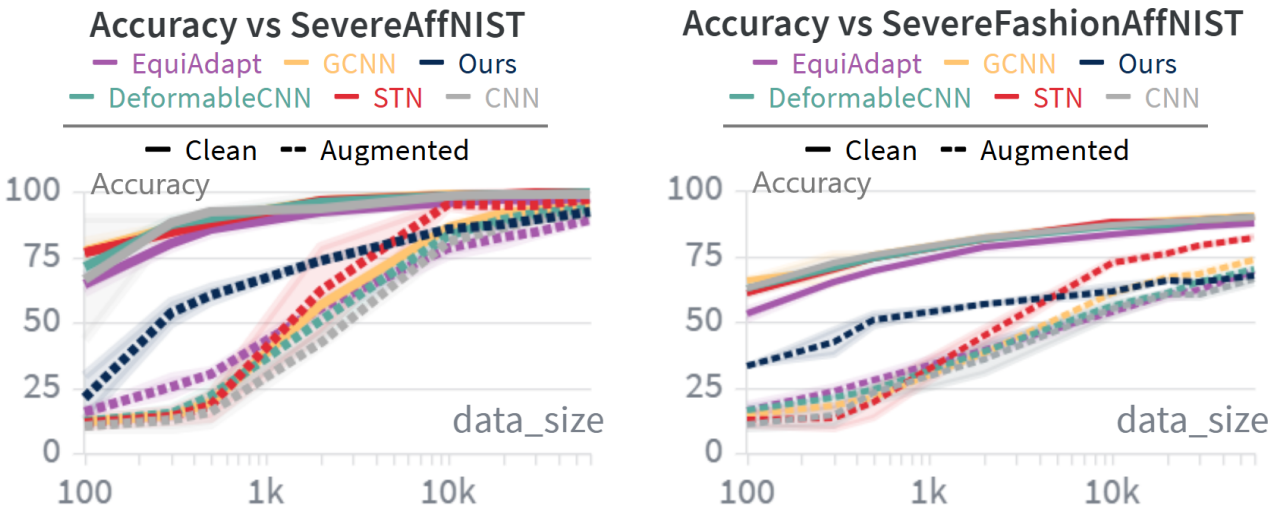

Figure 8. Results for competitors trained on clean (full) and augmented (dashed), with Ours, AxFuncta, having been trained and tested on Augmented. The goal for all methods trained on augmented is to match the accuracy obtained by methods on clean. Results showcase that our method (Ours), outperforms existing baselines on smaller datasizes up to around 2k. Left: SevereAffNIST. Right: SevereFashion-AffNIST. Performance measured through % accuracy across 5 runs. Datapoints at [100, 300, 500, 2000, 10000, 20000, 30000, 60000]

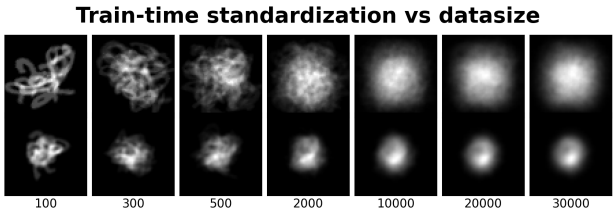

Figure 9. Train-time standardization on the stronger SevereAffNIST dataset across evaluated datasizes [100, 300, 500, 2000, 10000, 20000, 30000]. The procedure results in reduced variation, with separated digits stacking together.

see if our method can get as close as possible to the upper bound, despite having been given the augmented dataset. We perform our experiments on the Severe variants of our datasets as per [37], allowing AxFuncta to utilize its strengths as shown in the previous experiment. Dataset sizes range from only 100 samples (10 per digit), to 60000, split [0.8, 0.2] uniformly random to create a train and validation split. The test dataset is fixed to 10000 samples across all experiments. Experiments were run for 5 runs per datapoint, with the std dev being indicated. Results are shown in Figure 8 for both the SevereAffNIST and SevereFashionAffNIST datasets.

The shown figures indicate the problem to solve: When given a limited amount of data, existing baselines may lose out on significant amounts of accuracy when presented with geometric variation. This is most evident at the lowest data sizes, where methods such as the original CNN lose out on up to 64.98% raw accuracy when trained and tested on the SevereAffNIST dataset instead of the geometrically consistent MNIST. However, our method, AxFuncta, manages to outperform existing baselines in low-data regimes up to the evaluated datasize of 2000, enabling improved performance. However, beyond this point baselines such as the STN catch up and start outperforming our method on the augmented dataset. As such, our method is not recommended when working with large amounts of data, especially if newly presented samples are always expected to be in distribution.

To showcase the effectiveness of our train-time standardization approach, we once again display the mean sample, here before and after standardization, in Figure 9. Here, we showcase these results across our previously discussed set of dataset sizes. For the sake of clarity, the standardization is shown for only a single class, namely the digit '2'.

As can be seen, our train-time standardization procedure proves to be effective across all datasizes, with minimal differences being visible across the board. In terms of weaknesses, the lowest datasizes do see weaker alignment results, though this is expected given that there are few samples for the class-wise alignment part of our loss function to be effective. In turn, this is a likely contributor to the larger gap to our "clean" baselines for small data sizes. Notably, the overall end result is reminiscent of the 'Before' samples that can be seen towards the higher end of our Rotate Severity showcase in Figure 7. This overall alignment quality enables our fit process to take place, bringing about the strong results that we see across lower dataset sizes.

Ablation	OOD AffNIST	OOD FashionAffNIST
Base CNN	20.09%	26.01%
Bilinear sampler	61.80%	45.73%
No padding	80.03%	53.1%
No latent regularization	55.8%	42.65%
OOD AxFuncta	**83.85%**	**57.05%**

Table 2. An ablation performed on the OOD setting. CNN indicates performance without using AxFuncta. Bilinear sampler removes the use of the linearized sampler. No padding removes additional padding from the fitted images. No latent regularization removes the L2 regularization component from the loss function. Results indicate that all components are required for strong OOD performance. Ablation averages across 2 runs.

4.4. Ablations

To ensure that every element is required for proper use, an ablation test was performed on AxFuncta's fit procedure. This is performed on the same OOD setting as Experiment 1, found in subsection 4.1. We showcase our results in Table 2.

Here, our OOD results indicate that all components are required to obtain strong OOD generalization performance when using AxFuncta. Particularly, latent regularization, a core component, is shown to provide strong benefits as expected, resulting in a 28.05% improvement in raw accuracy on AffNIST, and 14.40% on FashionAffNIST. Further, the use of the linearized sampler over its bilinear counterpart results in a 24.23% increase on the AffNIST dataset, and 10.45% on FashionAffNIST, with padding providing a small additional benefit.

5. Conclusion and Future Work

In this work, a novel approach for 'fit' instead of 'predict' was presented, allowing models to focus on a simplified version of datasets with significantly less geometric variation. This is in an effort to bring about models that require less data for equivalent performance, while improving robustness to OOD samples. By training a generalizable INR based on Data to Functa logic to reconstruct our train set samples, new samples containing geometric variation can be moved towards a learned variable template through the use of a jointly optimized learnable affine vector. Through experiments we have shown that this provides increased accuracy in low-data regimes, though underperforms compared to existing alternatives when data is sufficient. Beyond this our method enables strong OOD performance on Affnist and FashionAffnist, a FashionMNIST counterpart. We demonstrate that our 'fit' logic is especially powerful in terms of translation and scale, showcasing strong robustness, though tends to struggle when faced with rotations in its current state. As such, we recommend leaving care of rotations with methods such as the EquiAdapt approach by Kaba et al. [17].

To the best of our knowledge we are the first work that uses generalizable INR networks in this fashion. Potential future work focuses on incorporating more strengths of generalizable INRs, such as the extension to other data modalities, as well as research on more compact classification pipelines by classifying Functa latents directly using our logic or improving incorporation of coordinate space transformations. Further work is also required in improving its performance when faced with rotations, as well as improving overall performance across larger datasizes.

Acknowledgements

We would like to thank Soumen Sinha for providing the SevereAffNIST and SevereFashionAffNIST datasets that were also used in their own experiments.

References

[1] Roberto Annunziata, Christos Sagonas, and Jacques Calì. Destnet: Densely fused spatial transformer networks, 2018. 2

[2] Roberto Annunziata, Christos Sagonas, and Jacques Cali. Jointly aligning millions of images with deep penalised reconstruction congealing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 81–90, 2019. 4, 1

[3] Roberto Annunziata, Christos Sagonas, and Jacques Cali. Jointly aligning millions of images with deep penalised reconstruction congealing, 2019. 2, 1

[4] Matthias Bauer, Emilien Dupont, Andy Brock, Dan Rosenbaum, Jonathan Richard Schwarz, and Hyunjik Kim. Spatial functa: Scaling functa to imagenet classification and generation, 2023. 3

[5] Nam Cao and Olga Saukh. Geometric data augmentations to mitigate distribution shifts in pollen classification from microscopic images, 2023. 2

[6] Yinbo Chen and Xiaolong Wang. Transformers as meta-learners for implicit neural representations, 2022. 3

[7] Taco S. Cohen and Max Welling. Group equivariant convolutional networks, 2016. 3, 5, 1

[8] Mark Cox, Sridha Sridharan, Simon Lucey, and Jeffrey Cohn. Least squares congealing for unsupervised alignment of images. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008:1–8, 2008. 2

[9] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks, 2017. 2, 5

[10] Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one, 2022. 2, 3, 4

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017. 3

[12] Lukas Finnveden, Ylva Jansson, and Tony Lindeberg. Understanding when spatial transformer networks do not support invariance, and what to do about it. In *2020 25th International Conference on Pattern Recognition (ICPR)*, page 3427–3434. IEEE, 2021. 2

[13] Alexander Gielisse and Jan van Gemert. End-to-end implicit neural representations for classification, 2025. 3

[14] Juan Eugenio Iglesias, Marc Modat, Loïc Peter, Allison Stevens, Roberto Annunziata, Tom Vercauteren, Ed Lein, Bruce Fischl, and Sebastien Ourselin. Joint registration and synthesis using a probabilistic model for alignment of mri and histological sections. *Medical Image Analysis*, 50:127–144, 2018. 2

[15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015. 1, 2, 5

[16] Wei Jiang, Weiwei Sun, Andrea Tagliasacchi, Eduard Trulls, and Kwang Moo Yi. Linearized multi-sampling for differentiable image transformation, 2019. 2, 4

[17] Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions, 2023. 2, 3, 5, 9

[18] Chiheon Kim, Doyup Lee, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Generalizable implicit neural representations via instance pattern composers, 2023. 3

[19] E.G. Learned-Miller. Data driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):236–250, 2006. 2

[20] Yann Lecun, Patrick Haffner, and Leon Bottou. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 1998. 5

[21] Hoosang Lee and Jeha Ryu. Toward efficient generalization in 3d human pose estimation via a canonical domain approach, 2025. 2

[22] Chen-Hsuan Lin and Simon Lucey. Inverse compositional spatial transformer networks, 2016. 2

[23] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields, 2021. 3

[24] Xiaoming Liu, Yan Tong, and Frederick W. Wheeler. Simultaneous alignment and clustering for an image ensemble. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1327–1334, 2009. 2

[25] Joost Luijmes, Alexander Gielisse, Roman Knyazhitskiy, and Jan van Gemert. Arc: Anchored representation clouds for high-resolution inr classification, 2025. 3

[26] George Ma, Yifei Wang, Derek Lim, Stefanie Jegelka, and Yisen Wang. A canonicalization perspective on invariant and equivariant learning, 2024. 2

[27] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 3

[28] Arnab Kumar Mondal, Siba Smarak Panigrahi, Sékou-Oumar Kaba, Sai Rajeswar, and Siamak Ravanbakhsh. Equivariant adaptation of large pretrained models, 2023. 3

[29] Tom Monnier, Thibault Groueix, and Mathieu Aubry. Deep transformation-invariant clustering, 2020. 2

[30] Dolev Ofri-Amar, Michal Geyer, Yoni Kasten, and Tali Dekel. Neural congealing: Aligning images to a joint semantic atlas, 2023. 2

[31] William Peebles, Jun-Yan Zhu, Richard Zhang, Antonio Torralba, Alexei A. Efros, and Eli Shechtman. Gan-supervised dense visual alignment, 2022. 2

[32] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018. 3

[33] Johann Schmidt and Sebastian Stober. Geometrically constrained and token-based probabilistic spatial transformers, 2025. 2

[34] Pola Schwöbel, Frederik Warburg, Martin Jørgensen, Kristoffer H. Madsen, and Søren Hauberg. Probabilistic spatial transformer networks, 2022. 2

[35] Zakhar Shumaylov, Peter Zaika, James Rowbottom, Ferdia Sherry, Melanie Weber, and Carola-Bibiane Schönlieb. Lie algebra canonicalization: Equivariant neural operators under arbitrary lie groups, 2025. 2

[36] Utkarsh Singhal, Ryan Feng, Stella X. Yu, and Atul Prakash. Test-time canonicalization by foundation models for robust perception, 2025. 2, 3

[37] Soumen Sinha, Jan van Gemert, and Alexander Gielisse. Are accidental scene conditions learnable?, 2026. 1, 2, 5, 8

[38] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020. 3

[39] Robin Winter, Marco Bertolini, Tuan Le, Frank Noé, and Djork-Arné Clevert. Unsupervised learning of group invariant and equivariant representations, 2024. 2

[40] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention, 2022. 2

[41] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 5

[42] Songxiao Yang, Yafei Ou, and Masatoshi Okutomi. Pixelinr: Scan-specific self-supervised mri reconstruction based on implicit neural representations. *Biomedical Signal Processing and Control*, 112:108838, 2026. 3

[43] Xiangrui Zeng, Gregory Howe, and Min Xu. End-to-end robust joint unsupervised image alignment and clustering. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3834–3846, 2021. 2

[44] Letian Zhang, Ming Li, Chen Chen, and Jie Xu. Il-nerf: Incremental learning for neural radiance fields with camera pose alignment, 2023. 3

[45] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *European conference on computer vision*, pages 566–583. Springer, 2020. 2

AxFuncta - Fit instead of Predict for Accidental Scene Conditions.

Supplementary Material

A. Experimental Setup

A.1. Functa model

For the Functa model, at train-time we closely follow the work of Dupont et al., reimplementing it to work in PyTorch 2.6.0. Based on the default values of the original paper, we utilize an INR model with sin activation functions, containing a depth of 10 and a width of 128 and a $w0$ of 30, with images being reconstructed over 4 inner-loop steps using MetaSGD. The image latents are 128-dim for MNIST and 256-dim for FashionMNIST. The model is trained using Model-Agnostic Meta-Learning for 2e4 iterations using the AdamW optimizer with an lr of 7e-7 and a batch size of 48. 2e4 iterations aligns with 20 epochs for a training set size of 48000. Due to issues with sudden representational collapses during the training of the INR, stability was further supported by using lowered betas in the AdamW optimizer: (0.85, 0.985).

At test-time, the above-trained model is reused, though the inner-loop process changes. Instead of performing 4 inner-loop steps, images are now reconstructed over 400 steps. While reconstructing these images, the latent is regularized using L2 regularization with a weight of 40.0. In the meantime, an affine operation is applied directly to the reconstructed image. The learning rate for gradient updates towards the affine is 50.0. These parameters were hyperparameter-tuned to be as high as possible without losing the image during reconstruction, enabling fast convergence. The inverse of the obtained affine is applied to the original image, which is then sent to the downstream classifier.

A.2. Classifier

The classifier architecture is shown in Figure 10. This architecture is trained for 4e4 iterations using the AdamW optimizer with an lr tuned [1e-2, 1e-3, 1e-4] and a batch size of 48. 4e4 iterations aligns with 40 epochs for a training set size of 48000. The loss function is Cross-Entropy loss. The CNN is trained separately from the Functa model. The same classifier is used universally for all methods and baselines. This forces all methods to work under the same constrained setting, putting the focus on the effects of the methods themselves rather than their classifier heads. The only exception to this is the GCNN for which the number of channels is halved as per the original paper [7].

B. Qualititative showcase OOD fit

Additional examples on individual samples are provided here. we ran AxFuncta over the course of 400 steps on the MNIST $\rightarrow$ SevereAffNIST setting, and plotted the original image, the reconstruction, and the transformed image. A qualitative showcase of our fitting approach can be seen in Figure 11.

Starting with the MNIST dataset, we can see that most digits are translated accordingly, with some rotations having been improved as well, especially for the '0' digits, which move from a horizontal alignment towards the expected vertical one, showcasing that our method is able to apply rotations of up to 90 degrees under the right conditions. However, there are also cases such as the number 6 at the top, where the reconstruction has only managed to reconstruct the circle placed in the center, as if it is reconstructing a '0' instead of a '6'. This showcases that our method can get stuck in local minima, which is suboptimal.

The FashionMNIST dataset sees correct translations, with clothes being scaled to mostly equal sizes. However, rotations seem to be an issue in this dataset. While some items such as the shirts at the bottom are able to be rotated to a nearly correct vertical orientation, most items have not rotated at all, while the shoe in the third row has rotated away from the expected canonical orientation. Of note is that the incorrect rotations are performed on images that did not end up with an accurate reconstruction, potentially indicating a local optimum.

C. Alternative training set standardization method

Rather than incorporating congealing work into our train-time method, it is also possible to utilize existing congealing work as a pre-processing step to our dataset. This in turn decouples the standardization process from the INR training process and therefore enables our INR to focus on a "standardized" version of our dataset, allowing us to utilize purely our base AxFuncta approach. To this end, we once again take advantage of existing work on image congealing by Annunziata et al. [2]. Our usage of this congealing method is further described in subsection C.1, with an experiment utilizing this approach shown in subsection C.2.

C.1. Congealing Setup

Before sending the training and validation sets to other models, their samples are congealed using existing work by Annunziata et al. [3]. Compared to the existing method, the main change is a larger base DeSTNet, resulting in the

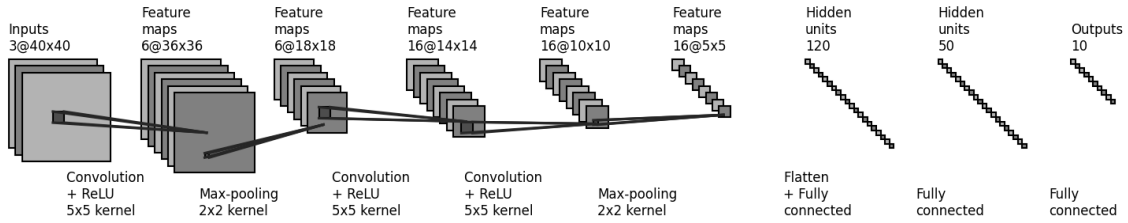

Figure 10. The shared classifier architecture.

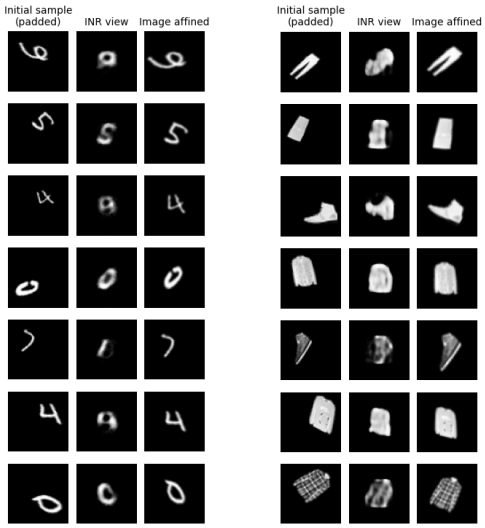

Figure 11. Qualitative showcase of OOD fit performance after 400 steps. Left: MNIST dataset, Right: FashionMNIST dataset. 3 columns left to right: Initial sample, the INR's current reconstruction, the transformed initial sample.

shape $F\{[$ conv7-4 — conv7-8 — conv7-8 — conv3-8 — conv1-8 $] \times 4\}$, where convD1-D2 indicates a D1xD1 receptive field with D2 channels, as per the original paper. The first three convolutional layers have a stride of 2. This is then combined with a padding of 6 to all images and a small squared affine loss, preventing the STN from learning to reconstruct all black by removing the samples through translation or shrinking. The congealing process is performed across 1e4 iterations per class. As the congealing operation did not operate as expected before the auto-encoder was able to reconstruct the image, the auto-encoder is warm-started for the first 40% of iterations.

The implemented changes accommodate for the larger affine transformations that are performed on the SevereAffNIST and SevereFashionAffNIST datasets compared to the original AffNIST dataset. This results in samples taking up much less space, leading to mostly black images. The implemented changes are made purely to strengthen the congealing process on the used datasets. We do not propose a new congealing method nor make any claims about any overall improvements.

C.2. Qualitative results Congeal+AxFuncta

The congealing approach is focused on tackling datasets that already contain geometric variation, frequently found in real world settings. To evaluate this approach, we therefore perform tests by training and testing on the full SevereAffNIST and SevereFashionAffNIST datasets. The performances of all methods are evaluated in terms of % accuracy, with the results showcased in Table 3.

Model	SevereAffNIST	SevereFashionAffNIST
CNN	91.43%	65.98%
STN	**97.27%**	**82.14%**
DeformableCNN	93.16%	69.76%
AxFuncta	91.92%	67.39%
Congeal+AxFuncta	39.44% $\pm$ 2.03	24.56% $\pm$ 1.58

Table 3. Accuracy obtained on the augmented $\rightarrow$ augmented setting.

As is shown in the figure, the attempted congealing approach underperforms significantly compared to all previously evaluated methods, including our full AxFuncta approach. This is interesting, as the congealing approach itself does seem effective, as per Figure 12.

Figure 12. SevereAffNIST digits after applying an existing congealing method.

It is possible that the deviations in digit locations, due to congealing being done separately per class, is enough to

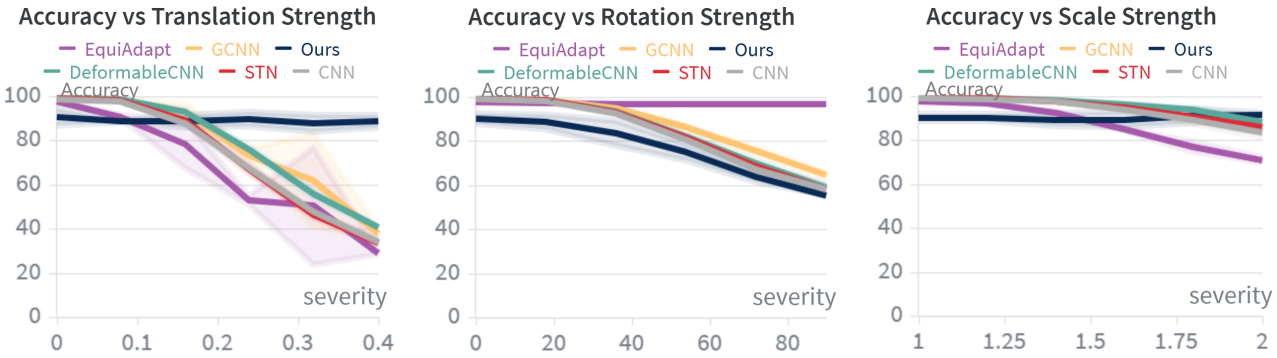

Figure 13. Experiment 2 performed on the MNIST dataset. Differences between baselines are more visible.

throw off the fitting process. This is also visible when comparing results to Figure 5. Possible solutions could be a multi-class variant of this congealing approach, or the incorporation of contrastive learning. Nevertheless, these are purely hypotheses. Due to time, and to not steer too far from the solution proposed in the main body of this report, this option was abandoned. Nevertheless, these may be interesting future directions, and are thus a challenge presented to the reader.

D. Experiment 2 baseline follow-up.

In experiment 2, an effect was observed where competitors had seemingly no advantage over one another. This seems strange, as one would expect some level of generalization to take place. However, FashionMNIST is a heavily standardized dataset, moreso than MNIST. As such, we provide additional results when the same experiment is performed on the MNIST dataset in Figure 13.

When performing the same experiment on MNIST, which has small amounts of variation within the dataset itself, the baselines start experiencing different results, with notably EquiAdapt being invariant to rotations. This shows that the level of generalization these baselines bring in OOD settings may be dependent on the level of variation present within the train set. Nonetheless, the overall results stay the same, with AxFuncta beating its competitors by being more robust to significant geometric variation in terms of rotations.

4

AI Statement

LLMs were used for the following use cases over the course of this research project.

- **Literature Search Support:** Summarizing/explaining parts of an existing research paper when it was unclear.
- **Idea Testing:** Confirming understanding of a topic within Computer Vision or a research paper by giving the LLM my perspective and asking for corrections.
- **Literature Search Support:** Asking about which sub-fields exist within a Computer Vision field.
- **Literature Search Support:** Confirming the existence of a topic within Computer Vision, i.e. confirming whether or not supervised congealing methods exist.
- **Extra:** Understanding how to use an HPC cluster, alongside its general functions, finding useful programs for it, the usage of Apptainer, as well as debugging some bugs that occurred through its usage.
- **Code Assistance:** The first iteration of invert_affine_2x3, a helper function which inverts an affine matrix. This was fully rewritten due to inaccuracies.
- **Code Assistance:** Implementation of the DeSTNet model, specifically named DeSTNet-4.