Stroke-Based Representation Learning - Final Report

Matthew Watson
Arizona State University
Group 15
mjwats10@asu.edu

Kaushal Rathi
Arizona State University
Group 15
krathi3@asu.edu

Hassan Ismaeel
Arizona State University
Group 15
hismaeel@asu.edu

Parth Kachhadia
Arizona State University
Group 15
pkachhad@asu.edu

Abstract

In this report we describe an approach to tackling the problem of spatial transformations for visual recognition models. Such models can suffer from poor performance at test time due to out-of-distribution test data if they are not trained on the appropriate spatial transformations. To combat this, our work explores using normalized Fourier transforms to yield affine transformation invariant representations of sketch strokes. We then train a variety of models on these representations for both MNIST and Google Quickdraw datasets, and compare to baseline models trained on the raw images. This report describes the implementation and experimental results of our proposed approach.

1 Introduction

The human vision system is able to recognize objects even when they are observed from an unusual perspective. For example, a car turned upside down or rotated in any other manner is still recognized as a car. Similarly, we recognize cars both when they are in the center of the fovea, and also when they are in our peripheral. These examples illustrate the robustness of human visual perception in the presence of rotation and translation applied to visual stimuli.

In contrast, computer vision models, such as deep neural networks, do not demonstrate the same robustness to these affine transformations. As a result, a model trained to recognize images of objects with a specific orientation may fail to recognize the same objects if the images are rotated or translated at test time. In many domains, this sensitivity to affine transformations is problematic because such transformations do not change the class label of the image.

This problem is widely noted in the literature, and is most commonly addressed through data augmentation at training time. To apply this technique successfully, one must anticipate the set of transformations the trained network might encounter during inference, and sample transformations from this set to apply to images at random during training. While this teaches the network to recognize images with these transformations applied, the network does not necessarily learn representations that are invariant to such transformations. Rather, it may simply memorize that transformed versions of images map to the same class as their untransformed counterparts.

In light of this, we wish to investigate the effects of using alternative representations which have invariance to affine transformation as an intrinsic property. Inspired by work on invariant representations done by Cheng et al. (2019), we explore the use of Fourier transforms of sketch contours to generate features invariant to affine transformations, which can be used as the inputs to a classifier in place of the raw sketch image.

We perform experiments using this technique on two datasets, MNIST and Google Quickdraw. On MNIST we apply our method to a single contour extracted from each digit image, with a Multilayer Perceptron (MLP) classifier, and compare to an MLP applied directly on the raw image, both with and without data augmentation during training.

For our experiments on Quickdraw, we use only the images from the Circle, Square, and Triangle classes. We perform the same experiment as on MNIST using the 28 x 28 rasterized reconstructions of the Quickdraw vector images. Additionally, we train a third model using the Quickdraw vector image binaries, which we call Fourier-GNN. Here, we identify and separately reconstruct each stroke in the image, extract a separate contour for each stroke, apply our Fourier method to each stroke contour, and then classify the resulting set of stroke descriptors using a Graph Convolution Network (GCN).

2 Implementation

To apply our method to a grayscale raster of a hand-drawn sketch, we first binarize the image by thresholding at an intensity value of 127 for images in the range [0, 255], or at a value of 0.5 for images in the range [0, 1]. We then use the OpenCV library to extract all exterior contours in the binarized image, keeping only the largest contour if there is more than one, see Fig. 1 for a visualization of image and external contour.

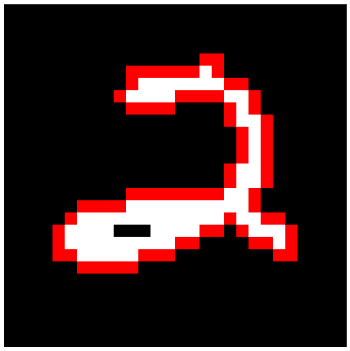

Figure 1: Binarized image of MNIST digit with contour extracted by OpenCV drawn in red

To apply a Fourier transform to the extracted contour, we adopt the method described by Kuhl & Giardina (1982), and implemented in the Python library Pyefd, which we shall now briefly explain.

Contours are expressed as closed loops, or chains, of adjacent pixels. We select a starting point on the contour, and as we complete one circuit clockwise around the loop, the resulting motion through 2D image space is projected onto the x-axis and y-axis.

Then, one Fourier transform is applied to the x-component of this motion, and another to the y-component. The Pyefd [4] library allows us to specify, via an ORDER parameter, how many terms, N, of the series to calculate, with a greater number of terms capturing more of the high-frequency details of the contour (Fig. 2). Both of the resulting truncated Fourier series have two coefficients for each of the N terms of the series (Fig. 3).

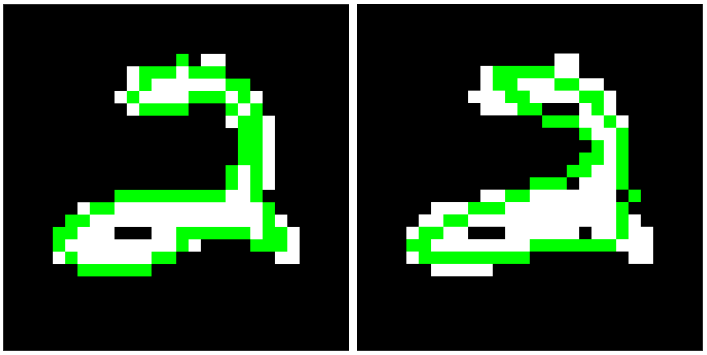

Figure 2: Left, Fourier order 7; right, Fourier order 3

$$X_N = A_0 + \sum_{n=1}^{N} a_n \cos\frac{2n\pi t}{T} + b_n \sin\frac{2n\pi t}{T},$$

$$Y_N = C_0 + \sum_{n=1}^{N} c_n \cos\frac{2n\pi t}{T} + d_n \sin\frac{2n\pi t}{T},$$

Figure 3: (Kuhl & Giardina, 1982) expressions for x and y-component Fourier series

The key step is to then *normalize* the resulting sets of Fourier series coefficients (Fourier descriptors) such that translations and rotations applied to the original contour do not change the *normalized* set of coefficients. To make the Fourier descriptors invariant to translation, we observe that the translation of the original contour affects only the first terms of each series, $X_0 = a_0$ and $Y_0 = c_0$. The point (a_0, c_0), known as the bias point, acts as the center point of the first harmonic phasor, and setting it to $(a_0 = 0, c_0 = 0)$ for all contours eliminates spatial position information.

Next, to make the Fourier descriptors invariant to rotation, we determine the angle θ between the image x-axis and the major axis of the first elliptical phasor. We then apply a rotation matrix to the contour to rotate it so that the image axes are now aligned with the major and minor axes of the first elliptical phasor. Intuitively, this step boils down to adopting the convention of rotating the contour so that the major axis of the first elliptical phasor aligns with the image x-axis. Thus, the values of the Fourier descriptors are a function of only the shape of the contour, not its rotation.

When this is complete, we end up with a feature array of size N x 4, where N is the number of terms of the Fourier series we are using, with each term $0 \leq n < N$ yielding four normalized coefficients (a_n, b_n, c_n, d_n). For MNIST and Quickdraw raster images, we train an MLP on these coefficient arrays extracted from the largest outer contour obtained from each image, and compare to the same MLP trained on the raw raster images.

Our Fourier-GNN, trained on Quickdraw vector images, leverages the fact that the vector images specify each stroke in the image separately to extract a separate Fourier coefficient array for each stroke in the image. Specifically, for a vector image containing k strokes, we first reconstruct each stroke in its own "sub-image" and extract Fourier descriptors for each stroke / sub-image as described previously.

Then we construct an adjacency matrix describing which strokes are connected to each other. This allows us to represent all the sketch information from each Quickdraw image as a graph, where each stroke is represented by its array of Fourier descriptors, and edges are placed between nodes whose corresponding strokes are connected in the image. We then train a GCN to classify this graph to obtain a classification for the vector image.

3 Experiments

Table 1: Classification accuracy for MLP on raw images with and without training on rotations and translations, MLP on Fourier descriptors extracted from single contour, and Fourier-GNN on stroke graph of all image contours

		Raw-MLP		Fourier-MLP	Fourier-GNN
		Augmentations	No-augment		
MNIST		**93.9%**	54.7%	93.5%	--
Quickdraw	Raster	96.2%	68.2%	**96.3%**	--
	Vector	--	--	--	95.4%

On MNIST we trained three models, an MLP on raw images with random rotations and translations applied, an MLP on raw images without rotations and translations, and an MLP trained on Fourier descriptors. The raw-image models were evaluated on test data which had random rotations and translations applied. For training and testing, spatial transformations were applied according to Engstrom et al. (2019). Each image had a rotation of θ degrees and translation $(\Delta x, \Delta y)$ applied, where θ was uniformly sampled from the range [-30, 30], and both Δx and Δy were integers uniformly sampled from [-3, 3]. The axis of rotation was the center of the image.

The same MLP was used for both raw images and Fourier descriptors: a ReLU network with 3 hidden layers, each with 512 units. In all three experiments, the model was trained for 10 epochs, and then evaluated on the test data. For all Fourier models, we report results obtained using the coefficients from the first 5 terms of the series, as our ablation analysis showed no improvement from using additional coefficients.

We ran exactly the same set of experiments on the 28x28 Quickdraw rasters from classes Circle, Square, and Triangle; only the number of output neurons changed from the MNIST experiments due to the change in number of classes. Since Quickdraw does not have separate training and testing sets, we set aside 20% of the dataset for testing, selected at random, and trained on the remaining 20%.

Our final model was the Fourier-GNN, described in the previous section. Specifically, we used a small GCN with 3 hidden layers that had 64 channels each, followed by average pooling and a linear classifier. As in the other experiments, this model was trained for 10 epochs on the training set before being evaluated on the test set. While we expected to see better accuracy from this model compared to the Fourier MLP, it performed slightly worse (Table 1). We speculate that the GCN used may not have been large enough to make use of the additional information in its input representation.

4 Discussion and Conclusion

Our experiments show that the normalized Fourier descriptors, even with coefficients from as few as 5 terms of the series, preserve enough information about the original contours to allow small MLPs and GCNs with three hidden layers to accurately classify the sketch the contour was extracted from. Additionally, the normalized Fourier descriptors of a contour are the same, no matter how the contour is rotated or its location in the image.

This inherent rotation and translation invariance means that using this representation as the input to a classification model automatically makes the resulting model robust to these test-time augmentations, without having to anticipate and train against them beforehand. However, while this property is often useful, there may be instances where it is undesirable. In some cases, image class may change when certain rotations are applied, for example, applying a 180-degree rotation to a digit such as a “9”. In other cases, the relative positions of disconnected contours in an image may be important to classifying the image, but this position information is discarded when applying our method.

In conclusion, our experiments with QuickDraw and MNIST datasets have shown that training classifiers of various architectures on normalized Fourier descriptors yields models with classification accuracy highly competitive with, and even exceeding that of models trained on raw images. Furthermore, this approach builds in invariance to affine transformations without having to exhaustively train on all spatial transformations the model might encounter at test time.

Future work could extend this approach to 3D strokes, or add a method of segmenting raster images into strokes to allow Fourier-GNN to be applied to standard image datasets.

5 References

[1] Cheng, S. (2021) SSR-GNNs: Stroke-Based Sketch Representation with Graph Neural Networks. (Preprint)

[2] Kuhl, F.P. & Giardina, C.R. (1982) Elliptic Fourier Features of a Closed Contour. *Computer Graphics and Image Processing*, Vol 18, Issue 3, pp. 236-258. ISSN 0146-664X, https://doi.org/10.1016/0146-664X(82)90034-X.

[3] Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., Madry, A. (2019) Exploring the Landscape of Spatial Robustness. *International Conference on Machine Learning*. pp. 1802–1811

[4] Blidh, H. Pyefd. https://github.com/hbldh/pyefd