Comparison of Feature Extraction Methods on Free-hand Sketches

Linh Dinh[1], *Xingyu Wang*[1]

[1]University of Chicago

ldinh@uchicago.edu, xingyuwang@uchicago.edu

Abstract

Visual information perception remains a central task of machine learning models. Image recognition has had significant progress in the past decade, and yet actual human perception covers a wider range of visual information encoded in more abstract forms of presentation. Learning sketches could help machine improve its progress and capability on abstract image perception.

We investigated feature extraction methods on Google's Quick, Draw! dataset, using Principal Component Analysis, Self-Supervised Learning, and Variational Autoencoder models, and analyzed the performance of each method on the sketches classification task. In our experiments, we found that self-supervised learning, the state-of-art technique for real image classification, has the best performance in sketches recognition. Furthermore, under our models, the pixel format of sketches gives better classification accuracy than the corresponding stroke/temporal format. Based on our results, we suggest that future work can explore self-supervised learning techniques on sketches recognition tasks and investigate better ways to use the temporal stroke data of sketches.

Index Terms: sketch recognition, image recognition, self-supervised learning, pretext task, deep learning

1. Introduction

Our project aims to evaluate various unsupervised feature extraction techniques on the free-hand sketches Quick, Draw! dataset [1], through a downstream classification task. Classifying the category of hand-free sketches resembles the task of MNIST image classification; however, sketches are often highly abstract and subject to very different drawing styles, making it a much more difficult task: both at the representation learning step and at the classification step. There are various applications for understanding and classifying abstract images such as signature verification, sketch-based facial recognition, machine-generated arts, etc. And, in many of these settings, annotated supervisory data could be difficult to obtain. Therefore, unsupervised learning experiments using the Quick, Draw! dataset would provide highly useful insights for related applications and settings. Additionally, free-hand sketches data is interesting because it contains both pixel (static) and stroke (temporal) information. Therefore, we could also investigate how these 2 types of information perform in our representation learning and classification models.

2. Data Processing

The original Quick, Draw! data is in vector format. For each sketch, there is a list of strokes, and each stroke is formatted as $[[x_0, x_1, x_2, \cdots], [y_0, y_1, y_2, \cdots], [t_0, t_1, t_2, \cdots]]$, where x and y are pixel coordinates, and t is the time in milliseconds since the first pixel. The dataset also provides corresponding Numpy bitmap files for each sketch that are derived from the vector data. We used the provided Numpy bitmap files as our pixel data. We processed the vector data by first concatenating all strokes and then formatting each sketch as a list $[[\Delta x_1, \Delta y_1, p_1], [\Delta x_2, \Delta y_2, p_2], \cdots]$, where Δx_i denotes the displacement of x_{i-1} to x_i in the vector data, Δy_i the displacement from y_{i-1} to y_i, and $p_i = 1$ if the current pixel is the end of a stroke and $p_i = 0$ otherwise. Therefore, we have our pixel data, which are 28×28 bitmaps, and ink data, which are sequences of size = number of points $\times 3$.

3. Related Work

3.1. Self-Supervised Learning

Most studies on the Quick, Draw! dataset are based on supervised learning. Common tasks on the dataset include sketch recognition (or classification) and new sketch generation tasks. Existing techniques on supervised tasks include convolutional neural network [2], deep hashing [3], graph neural networks and transformers [4]. Studies have also been conducted on recognizing sketches using a training set of real images [5]. Given that self-supervised techniques have achieved state-of-the-art results in computer vision classification tasks, researchers have started exploring the potential of self-supervised learning on sketches dataset [6]. The paper used recognition of geometric deformation and rotation as the pretext tasks, and trained classifiers using the features learned on pretext tasks. The paper used textual convolution network (TCN) to study the ink data, which is sequential and temporal, and convolution neural network (CNN) to study the pixel data. Our self-supervised learning experiments were inspired by [6].

3.2. Variational Autoencoder

Given that the Quick, Draw! dataset comes in vector format of sequences of strokes, a standard earlier method for completing sketches or generating new sketches from this data is a Sequence-to-Sequence Variational Autoencoder (VAE) model where the encoder is a bidirectional recurrent neural network (RNN) and the decoder is an autoregressive RNN [1]. Afterward, there were newer models such as Generative Adversarial Network Variational Autoencoder (GAN-VAE) or reinforcement learning [7]. Our variational autoencoder experiments, an RNN-VAE model using only ink data and a CNN-VAE model using only pixel data, were inspired by [1] and [6].

4. Experiments

4.1. Setup

We selected a random and fixed set of image classes, including tent, zebra, string bean, etc. Throughout all experiments, we trained a representation learning model on a large set of unlabeled data (75,000 samples with 50 classes of sketches), extracted features from this model, and used them in a downstream classification task. The downstream classification task used a Linear Support Vector Machine (SVM) to classify 1,000 labeled samples into their correct categories (e.g., tent, zebra, string bean, etc.). Some light tuning on the regularization parameter (C = 0.01, 0.1, 1.0) was applied on the Linear SVM.

4.2. Basic representation learning

The representation learning model is a Principal Component Analysis (PCA) model on pixel data. After tuning, the optimal number of components is 50, which were the 50 features extracted and used in our downstream SVM classifier.

4.3. Self-supervised learning

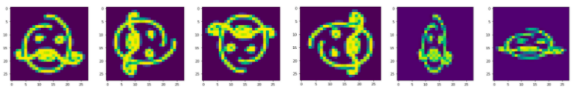


Figure 1: *Pixel Data Deformations*

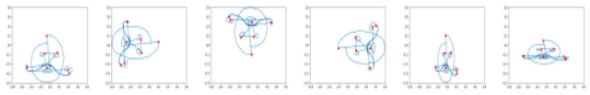


Figure 2: *Ink Data Deformations*

We designed two pretext tasks, rotation and compression. Then for both the ink and pixel data, we applied rotation (by 90, 180, and 270 degrees) and horizontal and vertical compression to each sketch (shown in Figure 1 and 2). Then we trained neural network classifiers to predict the type of geometric deformation applied given a transformed sketch. We trained the following two neural network models.

4.3.1. Ink data - RNN

We used a basic RNN architecture with linear layers which operate on an input and a hidden state, with a LogSoftmax layer after the output. We trained two RNN models to detect rotation and compression, respectively, using batch sizes of 1 and 100,000 iterations. Then, we used the last hidden layer of size 128 as our extracted features, which were used as input to the downstream SVM classifier.

4.3.2. Pixel data - CNN

We used a CNN architecture with 2 Conv2D layers with convolution filter size (3×3) and 16 and 32 output filtered arrays, two 2dMaxPool layers with filter size (2×2), ReLU activation function all over, and 2 fully-connected layers of sizes 800 and 64. Again, we trained two separate CNN models to detect rotation and compression, and took the last layer of size 64 as our extracted features for each sketch.

Since the two CNN models showed particularly good results, we concatenated both the features learned from rotation and compression, formed a vector of size 128, and used this vector as extracted features for an additional experiment.

4.4. Variational Encoder

4.4.1. Ink data - RNN-VAE

The representation learning model is an RNN-VAE on ink data. We used the open-source pre-train RNN-VAE model referenced in [1] with default hyper-parameters, and its encoder for feature extraction. In its default settings, the encoder is a bi-directional RNN with Long short-term memory (LSTM), and the decoder is an autoregressive mixture-density RNN with LSTM. The output dimension of the encoder is 128, which was also the 128 features used in our downstream SVM classifier.

4.4.2. Pixel data - CNN-VAE

The representation learning model is a CNN-VAE on pixel data. Our encoder has 2 Conv2D layers with kernel size (3x3) and stride (2). The input channels are 1 and 16, and the output channels are 16 and 32, for the first and second layer respectively. Both layers are activated with ReLU. This is then followed by a fully-connected layer and a Linear layer. The output dimension of this Linear layer is a hyper-parameter we tuned. The chosen output dimension, which is 64, was also the number of extracted features used in our downstream SVM classifier. Our decoder also has 2 ConvTranspose2D layers with kernel size (3x3) and stride (2). The input channels are 32 and 16, and the output channels are 16 and 1, for the first and second layer respectively. The first layer is activated with ReLU and the second layer is activated with Sigmoid. Some light tuning on learning rates and weights on Kullback–Leibler loss was also applied to the CNN-VAE.

5. Results

Feature Extraction Methods	**10 classes**	**50 classes**
No unlabeled data	40.3%	17.1%
PCA $n = 50$ components	55.3%	20.5%
Self-supervised rotation RNN	17.2%	4.4%
Self-supervised compression RNN	16.3%	3.5%
Self-supervised rot and comp RNN	17.2%	4.6%
Self-supervised rotation CNN	64.5%	26.5%
Self-supervised compression CNN	56.0%	19.2%
Self-supervised rot and comp CNN	66.6%	28.3%
Self-supervised rot RNN & CNN	64.7%	26.2%
Self-supervised comp RNN & CNN	55.6%	19.3%
VAE RNN	26.9%	3.9%
VAE CNN	43.8%	10.2%

Table 1: *Accuracy Rate Table*

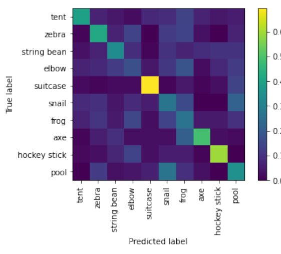
We constructed two test sets, each with 1,000 sketches. The first test set only contains 10 classes of sketches, and the second test set contains all 50 classes of images used in the feature extraction step. We then applied each feature extract model to the test set, and fed these features to a simple linear SVM, tuned on an unused dev set of 300 sketches, to predict the class of each sketch. The accuracy of each method is presented in Table 1.

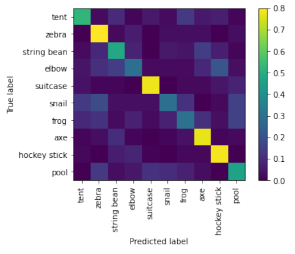
We observed that all of the models that used the ink data (i.e., the RNN models) did not provide us with satisfactory results. Also, predicting 50 classes of sketches is much harder than predicting 10 classes of sketches. However, for both 10 classes and 50 classes, the self-supervised model trained on rotation and compression pretext tasks with CNN provided us with the best results across all methods. The next close tie is the self-supervised model trained only on rotation pretext tasks.

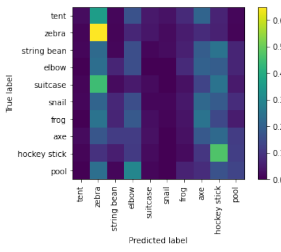
6. Discussion

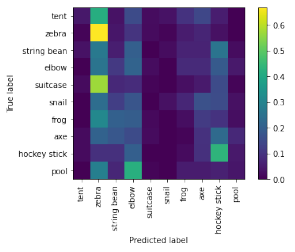
As seen in Results, a Linear PCA with relatively few dimensions as the representation learning model could provide a reasonable baseline accuracy on the downstream classification task that is as good as or often better than many other sophisticated neural network models we attempted. One hypothesis for this is the fact that, within a class of sketches (i.e., a zebra), there could be a lot of different styles and variations; thus, the neural network architectures probably attempted to capture and make sense of a lot of these nuances even though there might not be enough data to learn such nuanced patterns (i.e., users have very unique styles), resulting in spurious representations. Whereas Linear PCA probably attempted to learn the most general styles and shapes of each sketch category, which might be just enough to provide the SVM with the features it needs to do a reasonable, albeit not so great, job of classifying the sketch category.

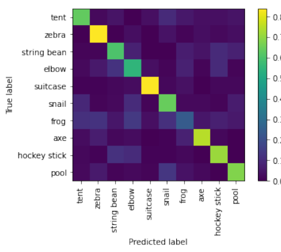
For each method, we plotted the confusion matrix for the 10-class prediction task to see which classes of sketches are frequently confused. The 10 classes are tent, zebra, string bean, elbow, suitcase, snail, frog, axe, hockey stick, and pool.


(a) *Labeled only*


(b) *PCA trained on unlabeled*


(c) *Rotation RNN*


(d) *Compression RNN*


(e) *Rotation CNN*

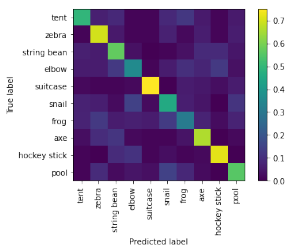
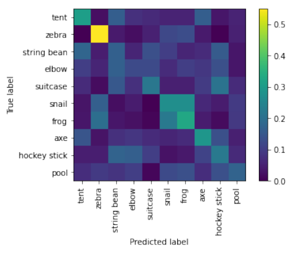

(f) *Compression CNN*

Figure 3: *Confusion matrices*

We found the following observations to be particularly in-


(g) *VAE RNN*

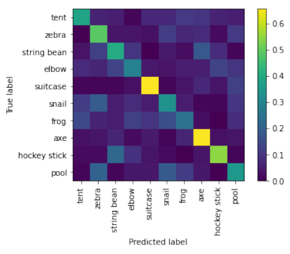
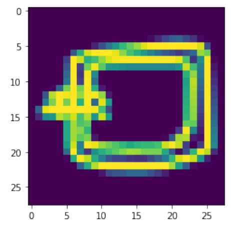

(h) *VAE CNN*

Figure 3: *Confusion matrices*

teresting and informative. Without applying any unsupervised learning techniques, Linear SVM is best at predicting suitcases. The reason for better prediction on the suitcase class could be the suitcase's unique square/rectangle shape, which is not shared by the rest of the other 9 classes. Meanwhile, snails and pools are the most frequently confused 2 classes. We examined the examples where the true label is pool and the predicted label is snail, and observed that these pool sketches tend to contain loops of circles, which makes the Linear SVM believe that these sketches are snails. Two examples are shown in Figure 4.


(a)

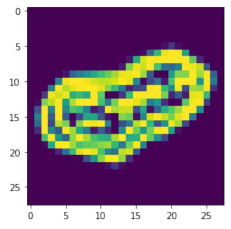

(b)

Figure 4: *Examples of a pool sketch confused with snail by without unlabeled data*

As soon as we apply the PCA model, we could observe an overall improvement in the prediction of all classes. As shown in Figure 3 (a) and (b), the scale of accuracy rates using the PCA model is noticeably higher than without it. Along with an overall accuracy improvement, PCA performed especially well on the classes of zebra, axe, and hockey stick. Meanwhile, the confusion between snail and pool weakened. We think that these improvements could be attributed to PCA capturing important information unique to each class, such as the head of an axe, stripes of a zebra, and the head of a snail.

Our self-supervised RNN model did not improve the accuracy of the classification task. We can observe that both the rotation and compression RNN models tend to characterize all images as only a few classes, such as zebra, elbow, and hockey stick. We are so far uncertain about the reason for the bad performances by RNN, but one possible explanation could be that our processed ink data only takes into account the displacement from one point to another. Even among the same classes of pictures, people tend to draw them differently and in various orders (clockwise or counterclockwise). Therefore, RNN might have a difficult time capturing pertinent classification information.

Our best accuracy rates were achieved on our CNN self-supervised models. We can observe that the confusion matrices of CNN models trained on the rotation pretext task are very similar to those trained on the compression pretext task, although the scale is different. Rotation CNN has higher overall accuracy than compression CNN. Compared to PCA, the self-supervised CNN models were particularly good at the classes string bean, elbow, snail, and pool. There was no particularly predominant confusion under these two models.

For our VAE models, we observed that just as in the self-supervised RNN model, zebra was predicted better than the rest of the classes. The rest of the confusion matrix of VAE RNN does not show any conclusive pattern. VAE CNN noticeably performed better than VAE RNN. Although the accuracy rate of VAE CNN did not improve upon the Linear SVM using learned features extracted from PCA, we noticed that the prediction of classes such as suitcase, axe, and hockey stick was better than others in VAE CNN. We think that these items tend to have less variation in their sketch styles, which might made it easier for the VAE to learn to reconstruct.

In general, from both our performance results and the confusion matrices, we observed that with our current experiment setups, pixel data was much more useful for the classification tasks. It is also important to notice that for the dataset on hand-drawing, since each class of drawings could be drastically different, any classification models can perform quite differently on different classes. In our experiments, we selected the 10 and 50 classes randomly (out of 346 total classes) and showed the results for both to check the robustness of our results. Lastly, by looking at confusion matrices and observing commonalities and differences across classes of sketches, we could hypothesize what type of information each feature extraction method is particularly good at capturing.

7. Future Work

We observed good results using self-supervised representation learning techniques on the pixel data in our experiments. Therefore, more extensive study on self-supervised learning techniques on the dataset could give even better results. One direction could be designing more complicated pretext tasks, such as more classes of geometric deformations, other than rotation and compression, or more complex rotation and compression transformations. Another direction could be experimenting with various types of neural network models in place of CNN.

Any of our models using the ink data had a poor performance. We suggest that alternative ways to process the original vector data could be explored. For example, instead of using the displacement $[\Delta x, \Delta y, p]$, one could simply use $[x, y, p]$ to capture more coordinate information. Other improvements could be done by applying more complex RNN or other neural network architectures to work with sequential data of variable lengths.

8. Conclusions

Learning a good representation for abstract images (i.e., free-hand sketches) to be used for a downstream classification task appears to be a quite challenging problem, reflected in our low accuracy rates across all experiments. However, the results from self-supervised learning techniques seem quite promising, suggesting that if we could design better (e.g., more difficult or more varied) pretext tasks, we might further improve the overall performance. To our surprise, Linear PCA performed particularly well given its level of complexity and can serve as a good baseline model.

In our experiments, the ink data models always underperformed the pixel data models and we did not succeed in utilizing the ink data to improve the accuracy of our models. However, we only attempted a simple version of RNN in these cases; thus, future work can explore more sophisticated sequence-to-sequence models with self-supervised learning techniques. We only quickly attempted to combine features extracted from the ink data models with features extracted from the pixel data models and did not see a performance improvement. However, past studies have tried this approach and have seen slightly better results. Therefore, this could also be an area for future work.

9. Acknowledgements

We would like to thank Professor Karen Livescu and Freda (Haoyue) Shi for teaching and TAing the course Unsupervised Learning and Data Analysis at TTIC, where we learned most of the techniques presented in this paper.

10. Additional Information

Python codes of all experiments can be found on this Github repo.

11. References

[1] D. Ha and D. Eck, “A Neural Representation of Sketch Drawings,” *arXiv e-prints*, p. arXiv:1704.03477, Apr. 2017.

[2] A. T. Kabakus, “A novel sketch recognition model based on convolutional neural networks,” in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. IEEE, Jun. 2020. [Online]. Available: https://doi.org/10.1109/hora49412.2020.9152911

[3] P. Xu, Y. Huang, T. Yuan, K. Pang, Y. Song, T. Xiang, T. M. Hospedales, Z. Ma, and J. Guo, “Sketchmate: Deep hashing for million-scale human sketch retrieval,” *CoRR*, vol. abs/1804.01401, 2018. [Online]. Available: http://arxiv.org/abs/1804.01401

[4] P. Xu, C. K. Joshi, and X. Bresson, “Multi-graph transformer for free-hand sketch recognition,” *CoRR*, vol. abs/1912.11258, 2019. [Online]. Available: http://arxiv.org/abs/1912.11258

[5] A. Lamb, S. Ozair, V. Verma, and D. Ha, “SketchTransfer: A challenging new task for exploring detail-invariance and the abstractions learned by deep networks,” in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2020. [Online]. Available: https://doi.org/10.1109/wacv45572.2020.9093327

[6] P. Xu, Z. Song, Q. Yin, Y.-Z. Song, and L. Wang, “Deep Self-Supervised Representation Learning for Free-Hand Sketch,” *arXiv e-prints*, p. arXiv:2002.00867, Feb. 2020.

[7] P. Xu, T. M. Hospedales, Q. Yin, Y.-Z. Song, T. Xiang, and L. Wang, “Deep Learning for Free-Hand Sketch: A Survey and A Toolbox,” *arXiv e-prints*, p. arXiv:2001.02600, Jan. 2020.