STAT 441 – Final Project Report

Hand-drawing Classification on "Google Quick, Draw!" Data

Yiqun Wang 20527280 | Renato Ferreira Pinto Junior 20501846 | Robert Huang 20543165 | Yufeng Wang 20552381

1 Introduction

1.1 Dataset

Quick, Draw! [1] is an online game developed by Google in which users are given one of 345 categories, and prompted to create an on-screen drawing corresponding to that category. As the user draws, the system has up to 20 seconds to guess the category. Over 50 million drawings of users have been collected and made available as an open-sourced dataset of 28×28 bitmaps.

1.2 Objective

The purpose of this project is to answer the question on the *Quick, Draw!* homepage: "Can a neural network learn to recognize doodling?" Convolutional neural network (CNN) is a class of models known for its robustness in image classification. One of the purposes of this project is to verify the effectiveness of CNNs in the hand-drawing classification problem when dealing with different sets of classes (categories). To be specific, we focus on two classification problems:

1. Classes that share a general structure;
2. Classes that are structurally different from each other.

We elaborate on this distinction in Section 2.2.

This analysis consists of two parts. First, we compare the performance of a CNN model with other models, namely feed-forward neural networks (FFNN) and boosted trees on a lower-dimensional representation. Second, we evaluate the ability of the CNN model to handle multi-class classification by comparing prediction accuracies against the number of classes involved in the two classification problems listed above.

[1] Available at `https://quickdraw.withgoogle.com/data`

In a different line of investigation, we attempt to modify the CNN model by integrating word embeddings of the class names into the model to explore potential improvements to classification accuracy.

Finally, we construct a *transfer learning* setting for this problem and explore how different models make use of structure in the data to perform well in the transfer learning task.

2 Experiments

We investigated how different types of machine learning models perform on the two classification problems. We implemented and trained a CNN model, a feed-forward neural network (FFNN) model, and an extreme gradient boosting tree model (XGBoost) [1].

2.1 Model architectures

The architecture of the CNN model has 3 convolutional layers. The first layer has 32 3×3 filters; the second layer has 64 3×3 filters; and the last layer has 128 3×3 filters. The activation function used in each convolutional layer is the Leaky ReLU [2], which we chose in an attempt to fix the problem of dying Rectified Linear Units (ReLUs) [3]. Additionally, after each convolutional layer there is a 2×2 max-pooling layer. Finally, the last dense layer uses the softmax activation function to output the results for the multi-class classification.

The architecture of the FFNN model has three hidden layers. Since input images are of size 28×28, we simply use 784 nodes in each layer. The activation function is again the Leaky ReLU for the same reason as above.

Finally, for the boosted tree model, we encountered a problem of having too many features with

weak predicting contribution, such as pixels at the corners of the picture. Therefore, we applied Principal Component Analysis (PCA) to the data to obtain the first 200 principal components as our input variable. The first 200 principal component could explain 90% of the variance of the original data and reduce the original dimension of 784 to 200. With the principal components as input, we trained the boosted tree model using XGBoost.

2.2 Datasets and Training

Recall that we study two classification problems: when classes are not generally similar to each other, and when classes share many features with each other.

For each classification problem, we selected 10 classes so that the length of the training process and complexity of the problem are acceptable yet challenging. Classes in the first problem are: *apple, basketball, circle, compass, cookie, donut, potato, soccer ball, watermelon, wheel*. Classes in the second problem are: *airplane, axe, bed, bicycle, butterfly, envelope, knife, square, star, donut*. Note that classes in the first set all share a "round" shape, whereas the ones in the second set do not have such common structure to the human eye. These two sets of classes will be referred to as "*Similar classes*" and "*Different classes*" for the rest of this report.

In training and evaluating models, we randomly split these datasets into 70% training set, 10% validation set and 20% test set. The random seed for this split was fixed across experiments.

The neural network models were optimized by the Adam optimization algorithm [4] and evaluated after each epoch on the validation set. The model with best validation accuracy was saved for final prediction. For the boosted tree model, we boosted 100 different trees with a learning rate of 0.1. Each tree has a maximum depth of 3, allowing it to have small predicting power while not being too complex.

2.3 Evaluation

We trained all models on the "Similar classes" and "Different classes" datasets; results are shown in Table 1. We can observe that the CNN model has the best performance in both classification problems among the three models, which confirms most findings in image classification.

For all three models, the test accuracy in the similar-class problem is significantly lower than the one in the different-class problem. This confirms our expectation that types of classes can affect performance of models strongly. Although the CNN model has the least decline in performance, it still fails to match its different-class performance in the similar-class task.

Model	Similar classes	Different classes
CNN	**86.800%**	**96.808%**
FFNN	80.898%	94.339%
XGBoost	62.066%	84.211%

Table 1: Performance of different models on the two classification problems

Another notable observation is that the FFNN model performs very well in the different-class problem, with a 2.5% lower test accuracy compared to the CNN model, whereas this difference is almost 6% in the different-class problem. One possible explanation for this result is that in the different-class case, images from different classes are different in enough locations that two-dimensional structure plays a less significant role in classification, allowing FFNNs to perform well. On the other hand, in the similar-class problem images with a round structure may share similar values in a number of positions of the vector; hence the FFNN, unable to exploit two-dimensional features in the way CNNs do, displays much poorer performance compared to the CNN.

For the boosted tree model, the most obvious observation is that there is a significant drop in the testing accuracy compared to FFNN and CNN. Besides that, we see that accuracy increases the most (22%) when changing the dataset to different classes. This is to be expected because boosted tree model was not able to capture the interactions among variables as well compared to Neutral Networks. Therefore, the structure of each input image has much higher effect on the prediction power. Since pictures in the similar class all have a round structure, the accuracy decreases drastically.

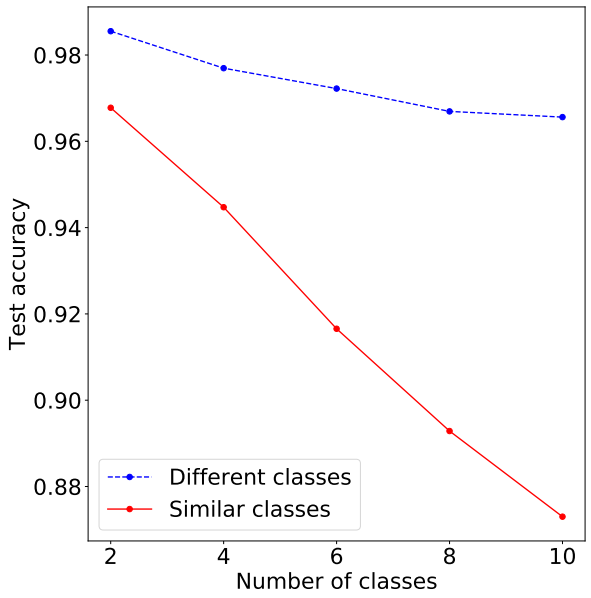
2.4 Accuracy vs. Number of classes

In order to investigate the effect of number of classes on the performance of the CNN model, we trained the model on random subsets of classes for each of the "Similar class" and "Different class" datasets. For 6 classes, 4 classes and 2 classes, we repeated the training process 10, 10 and 15 times respectively, and calculated the average test accuracy; this helps

ensure that we do not accidentally select a random "similar" subset from the "Different classes" dataset or vice versa since the number is small. The results are plotted in Figure 1.

There is a clear trend showing the decline in performance as the number of classes increased. This is to be expected, as a higher number of classes produces a raised likelihood of encountering confounding effects. The decline in accuracy is especially evident when modeling the "Similar classes" dataset, with a nearly 10A possible reason for this decline in performance is that the data in the "Similar classes" dataset share a lot of similar features, with lots of overlap between the classes present. This is not a significant issue when trying to classify a low number of classes, since there are other features that differ between the classes. However, when a higher number of classes are involved, the varying features are likely to overlap with each other, leading to confusion between those classes. On the other hand, the data in the "Different classes" dataset share much fewer features, and as a result is less affected by the increased number of classes.

Figure 1: Accuracy versus number of classes



2.5 Word embeddings

We also experimented with integrating word embeddings into our model. Intuitively, labels corresponding to different images in the data have specific meanings in English, and these meanings have relationships with each other. It is possible that a model that exploits this fact could learn more structure about the data. For example, "dog" and "cat" are related words in English, and drawings of the two animals ought to share many features. Hence, a model with prior knowledge about English label words may learn features relevant for cat classification even when it sees an image of a dog.

To test this hypothesis, we devised a model that attempts to fit images to their word embedding representations. Specifically, the model (which we call CNN-Emb) consists of the same convolutional layers as the CNN model, the same dense layer, and then, instead of connecting those units to a softmax classification layer, the model connects them to a vector of dimension e, where e is the dimension of the word embedding space. The objective, then, becomes a regression problem in this space, which we fit by minimizing mean squared error. For example, when the model sees an image of a dog, it tries to send its output vector close to the word embedding vector of "dog". At prediction time, the class closest in norm to the output vector in the embedding space is chosen as the output category.

In our experiments, we used 100-dimensional GloVe embeddings trained on Wikipedia and Gigaword data [5]. Contrary to our expectations, however, this model did not outperform the standard CNN model; rather, it consistently performed similarly or a small margin below CNN. We hypothesize that the information gained by the word embedding objective is not significant compared to the difficulty of modelling the data. Table 2 shows the results of our experiments.

Model	Similar classes	Different classes
CNN	**86.800%**	**96.808%**
CNN-Emb	86.169%	96.801%

Table 2: CNN model performance with and without word embeddings

2.6 Transfer learning

Since this dataset contains data for a large number of labels, it is possible to explore how the learning task for one set of labels relates to the learning task for a different set. In general, this process is known as *transfer learning* – using knowledge from one problem to solve a related problem.

To investigate how learning transfers among labels, we devised the following experiment: let S and T be two sets of labelled data such that the labels of examples in S are disjoint to those in T. We take all available examples from S, and train a model M on it. Then, we take only 1% of the examples in T – call the reduced set T' –, retrain M on T', and evaluate the test error on T. Assuming that the model can reuse its knowledge from S, this process should yield better results than simply training M on T' directly.

In our experiment, we took S and T to contain disjoint sets of 15 labels, with T' having only 1% of the training examples available to it. We evaluated models CNN and CNN-Emb on this transfer learning task. The CNN is retrained by reinitializing the final weights connecting the dense units to the output nodes (since these do not make sense for a different set of labels); on the other hand, CNN-Emb is retrained without dropping any weights, since the word embedding space does not depend on the set S or T. For comparison, we also evaluated the models after training only on T' (no transfer learning).

Table 3 shows the results from this experiment. As expected, transfer learning yields significantly better performance on T compared to training on T' alone. Moreover, CNN performs better than CNN-Emb. Together, these results seem to indicate that the most crucial learning occurs in the convolution layers, which capture general image features that are useful for classification in a wide range of classes. The final projection onto the specific classes, on the other hand, seems comparatively easier to learn as long as high-quality features are available.

Model	Accuracy on T
CNN with transfer	**86.2%**
CNN-Emb with transfer	84.4%
CNN without transfer	79.4%

Table 3: Performance of different models on transfer learning task

3 Conclusion

In this project, we have investigated how machine learning models can be trained to classify hand-drawn doodles into a number of categories. We explored how the difference or similarity between classes being considered affects the difficulty of the learning task; how difficulty increases as the number of classes considered by the same model grows; how semantic information about the English words used as labels can be added to a neural network model; and how models can make use of related data to transfer knowledge between tasks.

Among our findings were the fact that convolutional neural networks are some of the best performing models for this task, and that sets of labels for which images share significant structure make for a more challenging learning task across all models considered.

We also observed that although more classes are always harder to classify, the difficulty does not increase uniformly across categories – when the classes being considered are similar, the task becomes much harder as the number of classes grows compared to the case when all classes are very distinct from each other.

We verified that by introducing word embeddings as an alternative representation of class labels, and switching the learning objective to a regression objective, we can learn the classification task by considering the semantic properties of the English labels. However, this did not improve upon the CNN results, suggesting that CNN's powerful feature extraction properties are behind their strong performance.

Finally, we simulated a transfer learning task by evaluating a model on a different set of labels than its original training, after some fine-tuning on a very reduced set of new examples in the target set. This experiment showed that CNNs can learn features that translate well to other classes, and hence training on one set of labels can help the model predict a different set of labels.

We hope that this report has brought interesting considerations to light, and that the experiments and discussions shown here help expose some properties of the very challenging field of image classification.

References

[1] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016.

[2] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, p. 3, 2013.

[3] `https://cs231n.github.io/neural-networks-1/`.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.