Quick, Draw! Doodle Recognition Challenge

Aditya Agrawal
31221087
UMass Amherst
adityaagrawa@umass.edu

Arhum Savera
31267524
UMass Amherst
asavera@umass.edu

Abstract

Sketching or drawing has roots throughout the history of mankind- for simple representations of real-world entities- from Egyptian cave paintings to modern cartoonists. A major element of sketches is that they incorporate the creator's perception with a very strong flavor. The differences in RGB image classification versus drawing/doodle classification contribute towards the difficulty of this task, as doodles are usually grayscale. Issues with this task are further aggravated with the hugely varying perceptions of objects by humans. In this paper, we work on a sequential strokes-based doodle dataset collected by Google through the Quick Draw! Game, spanning across 340 classes. The dataset also includes incomplete & noisy doodles, with class labels. This work includes pre-processing and normalization of the dataset, where the doodles are of different sizes. We start off by experimenting with different baselines and state-of-the-art Convolution-based architectures models in the sketch recognition task, by pre-processing the sequential strokes data to 2-dimensional images data. We augment the implementation of the Convolution-based architecture with Recurrent layers- LSTM/Bidirectional-LSTMs, for capturing the temporal information in the strokes data and avoiding the preprocessing step of conversion into 2-dimensional images. We have also experimented with different initial learning rates, dropout rates, & callbacks for an improved learning process. We conclude with a successful implementation of a multi-label, multi-class classifier for a subset of classes in the dataset, with an accuracy of 84.18% and a top-3 accuracy of 93.63%. This work lays groundwork for future implementations spanning the entire dataset provided by Quick Draw! By Google.

1. Introduction

"Quick, Draw!" was released by Google as an experimental game to educate the public in a playful way about how AI works. The game prompts users to draw an image depicting a certain category, such as ”banana,” “table,” etc. We will be taking part in Google AI's Doodle Recognition challenge [1] hosted on Kaggle. This data is inherently noisy and requires building a classifier which can learn patterns from data which has been procured from the real world, submitted by real people.

The users are provided a freehand in drawing. Doodle recognition focuses on the intent with which the user is drawing, rather than modeling their unique, artistic styles. This foregoes the need to train the model on identifying distinctly styled sketches and vice-versa, for the user to freely draw without any formal training. Identification and classification of these sketches though still falls under the purview of image classification, which is a well-documented problem in the computer vision community. Grayscale nature of the sketches poses a further challenge, when compared to colored-images classification, where we have more information per pixel in the form of channels. Further, the simplistic nature of drawings and hugely diverse visual perception of objects by humans adds to the dilemma. Another issue with drawing classification is the availability of large blank spaces throughout as compared to complete RGB photographs, which in a way can also help by allowing compression of the training data.

Lastly, the sources of variation in both of these types of images- sketches and photographs- is also very different. Photographs have an impact from occlusion, viewpoint variation, illumination conditions, scale variation, background clutter, and deformation, to name a few. As compared to sketches, where primary source of variation is the significantly different perception of the same object by different humans. This can also be termed as the difference in 'artistic styles'.

The available dataset contains 50 million drawings encompassing 340 label categories, given as a sequence of strokes in x, y co-ordinates. These categories include many real-world objects such as airplanes, lions, cake and ice cream. The task is to build a classifier for the doodles, which can perform well on a manually-labeled test set from a different distribution. This problem has applications in OCR, NLP and other recognition problems.

The following sections outline & detail our approach to

this problem, which includes a thorough literature survey for understanding the current status quo in Section 2. This is followed with the framework description and experimental details in Section 3 & 4, where we start off by applying a state-of-art convolutional architecture, a.k.a Sketch-A-Net [2] for static drawing images, after preprocessing the strokes data into 2-D images. This is followed with incorporation of the sequential information in data in the form of LSTM-based architectures and the final model involving a combination of Sketch-A-Net and LSTM layers.

2. Background/Related Work

2.1 *"How do Humans Sketch Objects?" [3]*

This work introduced a dataset of 20,000 sketches using Amazon Mechanical Turk spanning 250 categories. The authors developed bag-of-features sketch representation and used multi-class support vector machines in this work. They also demonstrated that bag-of-features representations built on gradient features could achieve reasonable classification accuracy (56%).

2.2 Sketch-Me-That-Shoe *[4]*

Sketch-Me-That-Shoe investigated the problem of fine-grained sketch-based image retrieval. They developed a deep triplet ranking model for instance-level SBIR with a novel data augmentation and staged pre-training strategy to alleviate the issue of insufficient fine-grained training data.

2.3 SketchNet *[5]*

SketchNet presented a weakly supervised approach that discovers the discriminative structures of sketch images, given pairs of sketch images and web images. In contrast to traditional approaches that use global appearance features or relay on keypoint features- they developed a triplet composed of sketch, positive and negative real image as the input of their CNN. Then a ranking mechanism is introduced to make the positive pairs obtain a higher score comparing over negative ones to achieve robust representation.

Index	Layer	Type	Filter Size	Filter Num	Stride	Pad	Output Size
0		Input	-	-	-	-	225×225
1	L1	Conv	15×15	64	3	0	71×71
2		ReLU	-	-	-	-	71×71
3		Maxpool	3×3	-	2	0	35×35
4	L2	Conv	5×5	128	1	0	31×31
5		ReLU	-	-	-	-	31×31
6		Maxpool	3×3	-	2	0	15×15
7	L3	Conv	3×3	256	1	1	15×15
8		ReLU	-	-	-	-	15×15
9	L4	Conv	3×3	256	1	1	15×15
10		ReLU	-	-	-	-	15×15
11	L5	Conv	3×3	256	1	1	15×15
12		ReLU	-	-	-	-	15×15
13		Maxpool	3×3	-	2	0	7×7
14	L6	Conv(=FC)	7×7	512	1	0	1×1
15		ReLU	-	-	-	-	1×1
16		Dropout (0.50)	-	-	-	-	1×1
17	L7	Conv(=FC)	1×1	512	1	0	1×1
18		ReLU	-	-	-	-	1×1
19		Dropout (0.50)	-	-	-	-	1×1
20	L8	Conv(=FC)	1×1	250	1	0	1×1

Table 1: The architecture of Sketch-a-Net.

2.4 Sketch-A-Net *[2]*

Sketch-a-Net [3] showed that deep features can surpass human recognition accuracy – 75% compared to the 73% accuracy from crowd workers in [2]. They introduced a CNN architecture specifically for sketch classification. Their architecture outperformed a variety of alternatives. These included the conventional HOG-SVM pipeline, structured ensemble matching, multi-kernel SVM, Fisher Vector Spatial Pooling (FV-SP), and DNN based models including AlexNet and LeNet.

3. Approach

Our main approach throughout this work is to tackle the problem from a deep learning perspective which was also justified by the results obtained on two classical machine learning algorithms- Support Vector Machines and K-Nearest Neighbors classifier. For the purpose of this experiment, we first preprocessed the sequential stroke data into 2-dimensional images. We observed that the parametric SVM classifier- which can be viewed as 1-layer Neural network- lacked sufficient capacity to model such a complicated dataset and resulted in a poor performance of 33% top-1 accuracy and it also did not provide us with probabilistic outputs needed for the top-3 accuracy evaluation metric. The non-parametric K-NN classifier gave best results with K set to 10, but did not surpass top-1 accuracy of 41% and a top-3 accuracy of 53%. It further led to an extortionate testing time due to the expensive computation of distances between the images. Using K-NN is also not feasible as it would not be scalable to the entire dataset of approx. 50 million images, as the training data needs to be available for every classification. This leads to the need of a high capacity model which allows us to discard the training data at test time and favors a faster testing time, thus- Deep Learning.

Our first approach was to implement the architecture of Sketch-a-Net CNN for our problem. The architecture is described in Table 1**.** Sketch-a-Net is a deep CNN. Despite all the efforts so far, it remains an open question how to design the architecture of CNNs given a specific visual recognition task; but recent recognition networks follow a design pattern of multiple convolutional layers followed by fully connected layers. The network uses convolutional layers, max pooling and ReLU activations for its convolution blocks. The final layer uses Softmax. After using Sketch-A-Net as our baseline, we look towards using a CNN-LSTM hybrid model for this problem. Our motivation for this approach is that Sketch-A-Net was designed for sketches which were readily available as images.

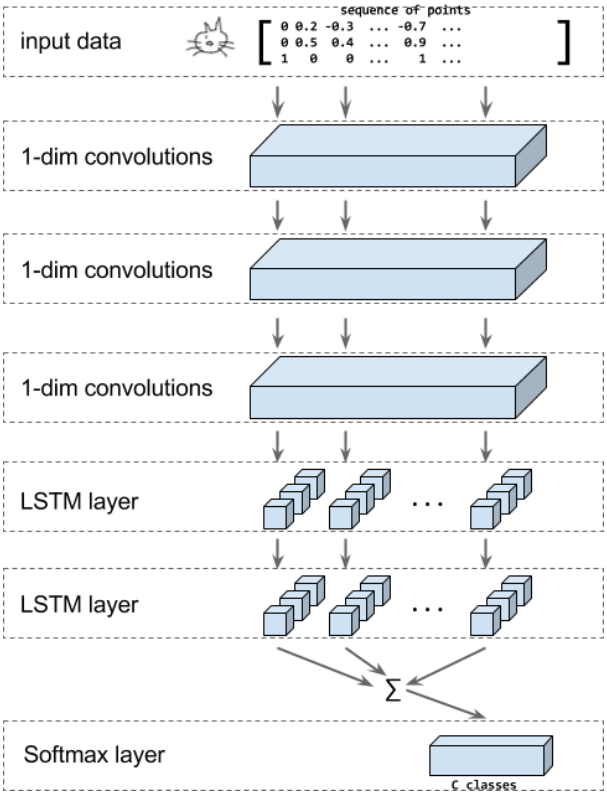

Fig. 1 CNN-LSTM Architecture.

Index	Layer	Filter Size	Filter Num
0	Batch Normalization	-	
1	Conv – ReLU - Maxpool	5x5	128
2	Conv – ReLU - Maxpool	3x3	256
3	Conv – ReLU - Maxpool	3x3	256
4	Conv – ReLU - Maxpool	3x3	256
5	LSTM/ BiLSTM	-	128
6	LSTM/BiLSTM	-	128
7	FC	-	512

Table 2. SAN-LSTM/ Sketch-A-BiLSTM. (Dropout applied after every layer, except 0.)

Our chosen dataset requires an extra processing step to convert the input to the required format. Doing so results in loss of vital sequential information. The sketches are originally available as strokes in coordinate space and contain temporal information which we can use to build our classifier. The nature of this data brings forth the use of a recurrent neural network. Our basic CNN-LSTM model can be visualized in Fig. 1.

We take our input data and transform it into three-dimensional data containing difference between subsequent co-ordinates in a stroke in the x & y dimension and an indicator for the row being a starting stroke of a new image. We use Batch-Normalization as the first layer, as a pre-processing step for zero-centering the data. We, further, use 1-dimensional convolutions as our feature extractors, which connect to the LSTM cells, whose outputs are then summed and fed to fully connected layers followed by a Softmax classification. We use LSTMs to avoid the vanishing gradient problem. [6] We use the Softmax classification function as the Doodle recognition challenge is essentially a classification problem. For classification problems we usually either use categorical cross entropy or hinge loss. Cross entropy loss minimization leads to well-behaved probabilistic outputs and Hinge loss doesn't help with probability estimation. Instead, it punishes misclassifications. As one of our metrics is Top 3 accuracy, using categorical cross entropy was the logical choice. We coupled this with a Softmax activation at the end of our network for probabilistic inference of the top 3 classes.

As our 1-dimensional convolutions act as a feature extractor, we then look to improve the capacity of the model. To do this, we take inspiration from Sketch-A-Net and use the design of the **L2, L3, L4, L5** (refer to Table 1) convolutional blocks in our CNN-LSTM (Referred to as SAN-LSTM in our results). Another variation on our CNN-LSTM model is to replace LSTM cells with Bidirectional-LSTMs as we expect getting past and future information can improve the performance. This variation is referred to as CNN-BiLSTM.

Finally, we present our final model, improving both components of our CNN-LSTM model by using the Sketch--A-Net convolution blocks as our feature extractor and Bidirectional LSTMs for our recurrent component- Sketch-A-BiLSTM in Table 2.

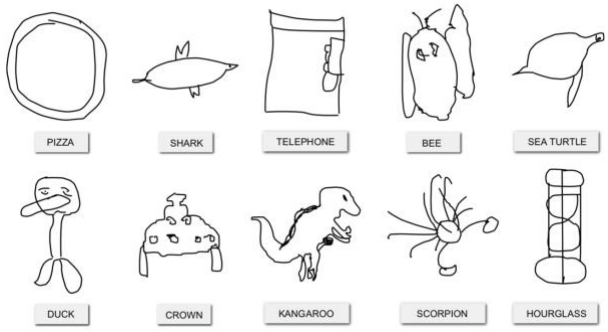

Fig. 2- Confusing doodles in the dataset.

4. Experiment

Experiments were conducted to classify 50 different classes out of the original 340. A random seed was set to ensure the classes sampled were the same across different experiments and models. A batch size of 2048 images was used with 3000 images from each class. The size of the test and validation set were both 750 images per class. Dropout was added throughout the model architecture to prevent overfitting.

To help expedite the training and hyper parameter selection process, we used the learning rate reduction callback available in the Tensorflow Keras API [7]. We reduced learning rate when our validation loss had stopped improving.

We set the initial learning rate to 0.0001 and let it be reduced by a factor of 0.4 whenever we reached a plateau. Xavier initialization was used wherever weights initialization was needed.

We also used early stopping to stop training our model when no improvement was seen over a set number of epochs. The model was trained with a categorical cross entropy loss and used an Adam optimizer. We chose Adam as it is generally regarded as being fairly robust to the choice of hyper parameters, though the learning rate sometimes needs to be changed from the suggested default. We can find the results of our experiments on the discussed models, in Section-3, using the Google Doodle dataset in Table 5.

In Table 5, we show the results of our experiments- where we have used 2 evaluation metrics:

1. Top 3 Accuracy- It is defined as the accuracy of the true class being in the top-3 predicted classes.
2. Top-1 Accuracy- It is defined as the accuracy if the correct class is predicted or not.

We can see that the default architecture of Sketch-A-Net gave us decent results on the images (converted from strokes data available to us)- achieving a 92.34% top-3 class prediction accuracy & 73.4% top-3 accuracy on the test set. Using CNN-LSTM architecture, shown in Fig. 1, improved the Top-1 accuracy from 73.4% to 81.1%, but the Top-3 accuracy was approximately the same. Using a Bidirectional LSTM within the same architecture further improved the Top-1 Test Accuracy (CNN-BiLSTM)- from 81.1% to 82.5%.

Interpreting the sketches as sequences using the recurrent models (CNN-LSTM & CNN-BiLSTM) resulted in a higher accuracy than converting the data as single channel images and using them as pixels with the CNN, which was done in Sketch-A-Net. One possible reason for this is the nature from which the data was gathered. The data is originally to be drawn within a certain time limit. This means that users are more likely to draw the important and

Learning Rate	Top-1 Accuracy
0.00001	80.86
0.00005	81.34
0.0001	84.18
0.0005	84.02
0.001	78.31

Table 3. Learning Rate vs. Top-1 Validation Accuracies.

Convolution Dropout	LSTM Dropout	Top-1 Accuracy
None	None	80.39
None	0.3	81.87
0.15	None	82.05
0.15	0.3	84.18
0.3	0.15	83.38
0.1	0.4	84.06
0.4	0.1	83.71

Table 4. Dropout Rates Experiment- Convolutional & LSTM.

Model	Test Accuracy (Top 1)	Test Accuracy (Top 3)
Sketch-A-Net	73.4	92.34
CNN-LSTM	81.1	92.3
SAN-LSTM	83.6	93.3
CNN-BiLSTM	82.5	93.3
Sketch-A-BiLSTM	84.18	93.63

Table 5. Results – Top-1 Accuracy vs Top-3 Accuracy.

significant strokes first which are more likely to indicate the main class of the sketch, and then follow up by adding the details later. Using this temporal information, we are able to build a more powerful classifier than using a purely CNN based architecture which loses this sequential information and only considers the final resulting image which may often be incomplete or noisy.

Finally, we used portions of the Sketch-A-Net architecture as a more powerful feature extractor than 1-dimensional convolutions for our LSTM & BiLSTM based models, which can be seen in Table 2. This resulted in further improvements in the top-1 accuracies- SAN-LSTM gave an improvement of 2.5% over the CNN-LSTM and Sketch-A-BiLSTM gave an improvement of 1.7% over CNN-BiLSTM.

We can refer to Table 3. to understand our choice of learning rate, where we have shown the validation accuracies achieved for a maximum of 100 epochs in comparison to the initial learning rates, for Sketch-A-BiLSTM architecture. We can see that very slow learning rates hinder the speed of the learning process, also triggering early stopping callback, and very high learning rates result in lowered validation accuracies. Thus, we decided to go with the optimum learning rate of 0.0001. Further, it is important to mention that we employed a decayed form of the learning rate.

Further, we experimented with dropout rates differently in Convolution layers and LSTM/BiLSTM layers, results of which are shown in Table 4. Experimenting with dropout was a crucial task as a good generalization performance is especially needed for our dataset, where each class can take up any form that spans people's imaginations of a doodle. We tried several different combinations – no dropout in convolution layers/ LSTM layers, to different rates in both the layers. We empirically noticed a pattern that dropping with a higher rate deeper in the network was more important and aided the validation accuracies and thus settled with dropout rates of 0.15 for the convolution layers and 0.3 for the deeper-lying LSTM/BiLSTM layers.

The lack of significant improvement in Top-1 Test Accuracy can be attributed to the data being noisy due to incomplete images, random doodles which are not supposed to mean anything and images which are confusing for any human observer as well. We can refer to these images from the Google's Quick Draw [1] in Fig. 2, where we can see that the Kangaroo could have been easily interpreted as a Dinosaur by many. We can observe that human errors are usually confusions between semantically similar categories (e.g. 'panda' and 'bear'), although geometric similarity accounts for some errors as well (e.g. 'tire' and 'donut'). One important thing to note here is that

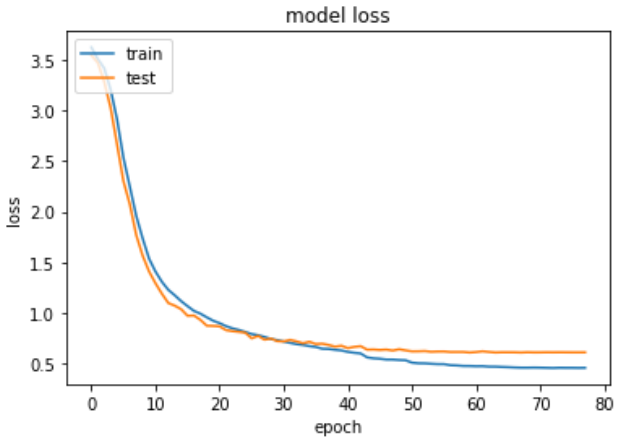

Fig. 3 CNN- LSTM Loss History.

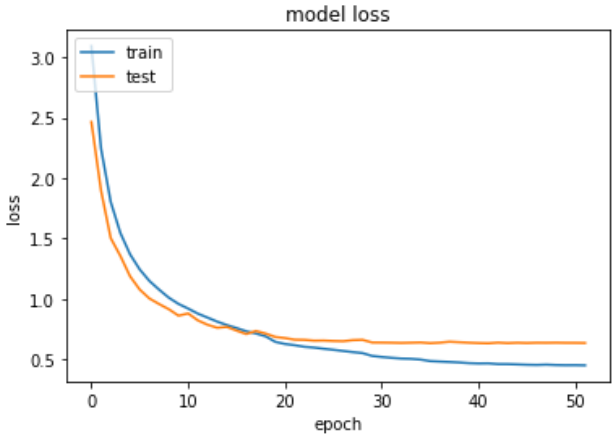

Fig. 4 CNN- Bi-directional LSTM Loss History.

the dataset contains class labels for all the images, even for completely noisy ones.

Another observation was that although replacing 1-dimensional convolutions with the Sketch-A-Net convolution blocks yielded better results, it also took a longer time to begin training accurately. This can be credited to the increased number of weights in our CNN component of our SAN-LSTM architecture. Similarly, when using bidirectional LSTM cells instead of unidirectional, we see high accuracies in the first few epochs, much faster than the CNN-LSTM model, which can be seen in Fig. 3 & Fig. 4. We can observe that at around 10 epochs, bidirectional LSTM loss had already dropped to approx. 0.8 while unidirectional-LSTM was at approx. 1.4. It, further, took 20 more epochs for unidirectional-LSTM to achieve the same loss as bidirectional-LSTM.

Further, we observe an empirical advantage by using

bidirectional-LSTMs in this setting as the early stopping takes place at approx. 50 epochs but the unidirectional-LSTM model takes almost 30 more epochs to achieve similar loss and also fails to perform as well as the bidirectional counterpart. It proves that using both directions in this type of sequence classification problem is a useful technique. We can also view the qualitative results in Fig. 5 & Fig. 6. We can see that the classifier does a good job at learning the general shape of objects, for example, it learns the shape and stripes of a bee in Fig. 5/ the shape of a lions' mane in Fig. 6.

Fig. 5 Bees in the dataset. (Green- correctly classified by final model, Red- Incorrect classification)

Fig. 6 Lions in the dataset. (Green- correctly classified by final model, Red- Incorrect classification)

Interestingly enough, the examples that were misclassified were hard to classify even for the authors- some doodles are just messy scribbles. This is due to the fact that this dataset is collected from the real world and thus is inherently noisy so there are bound to be sketches which make little to no sense. The classifier performs well in that it also learns to identify some form of body as well for the lion, considering the pre-dominance of mane in the data.

5. Conclusion

We have implemented a multi-class classifier for the classification of drawings for 50 classes, with a fairly good performance using recurrent neural networks and convolutions for feature extraction. Different architectures were experimented with, including bi-directional LSTMs- achieving an over 93% top-3 accuracy. In order to achieve a higher accuracy, the amount of training and evaluation data per category could be increased and the entire dataset could be leveraged for training a more accurate model. Another improvement could be to pre-process the data set by removing sketches which can be considered as noise. This could include use of Mechanical Turk to identify incomplete images and creating a separate category for these images. Removing unnecessary image processing in the implementation of the CNN might also be a way of improving the results, which would involve studying the contribution of different CNN layers towards the final output. Different architectures such as attention-based and residual neural networks could also be applied to this problem. We expect them to outperform simple CNNs.

References

[1] https://www.kaggle.com/c/quickdraw-doodle-recognition/data.

[2] A. Yu, Q., Yang, Y., Song, Y.-Z., Xiang, T., And Hospedales T. 2015. Sketch-a-net that beats humans. In British Machine Vision Conference (BMVC)., 2004.

[3] Eitz, M., Hays, J., And Alexa, M. 2012. How Do Humans Sketchobjects? Acmtrans.Graph.(Proc.Siggraph)31,4, 44:1–44:10.

[4] Yu, Q., Liu, F., Song, Y., Xiang, T., Hospedales, T., And Loy, C. C. 2016. Sketch me that shoe. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[5] H. Zhang, S. Liu, C. Zhang, W. Ren, R. Wang and X. Cao, "SketchNet: Sketch Classification with Web Images," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 1105-1113.

[6] H. J, "Long short-term memory. - PubMed - NCBI", Ncbi.nlm.nih.gov, 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/9377276. [Accessed: 18- Dec- 2018].

[7] "Keras | TensorFlow," TensorFlow. [Online]. Available: https://www.tensorflow.org/guide/keras. [Accessed: 18-Dec-2018].