Sketch ⇒ Label
How CNNs performance sketch classification?

Sushan Bhattarai
Stanford University
Stanford, CA
sushanb@stanford.edu

Abstract

Due to increase in touch devices, a lot of people draw sketches/outline for conveying the idea or message. Since text cannot express a lot of information and ideas, sketches act as an easy medium of expressing those ideas. In Computer vision, image classification problem is the fundamental base for every complex problems like object segmentation, object detection and so on. However, sparse amount of work has been done in classifying sketches or outlines of an image. Hence, we explored the idea of evaluating the state of art networks for classifying the sketches/outline of the objects instead of the real image.We trained a four layer convolutional network from scratch and fined tuned a pretrained Resnet50 and DenseNet. After training our model on 20k sketches images across 125 categories, we got test set top 1% accuracy of 35%, 82% and 77% for four layer ConvNet, fine tuned ResNet and fine tuned DenseNet respectively.

1. Introduction

We have seen exponential growth in the development of touch sensitive devices. As a result, people tend to do a lot of drawing of ideas and sketches to represent the things in the real world. It will be great if we can organize this abundance of the information by using the state of art image classification techniques. Sketch classification is an interesting problem to explore how neural networks thinks about the image when doing the classification. Generally, we have seen a lot of work done in image classification where the images are real world image taken by cameras and so on. However, sketches are gray scale or black and white images that are representative of objects.

Next thing why sketch classification is an interesting problem to explore is the presence of high intra class variation and low inter class variation. We can have different sketches of a same object. It means that variation within the same class is very wide. However, we have low inter class variation in sketches. For example: if we compare the sketch of ball and earth, it will be very similar. Infact it is very difficult for a normal human to achieve high accuracy in fine grain classification tasks. Since the sketch do not have texture and color as of real image, it becomes a complex problem.

The input to our algorithm is an image that is sketch of an object. We have 20000 images of the objects. The output of the algorithm is the predicted class of the sketch.We use convolutional neural network as the pipeline for the sketch classification which is described in detail in section 4.

2. Related Work

Early studies on sketch recognition worked with professional CAD or artistic drawings as input [7]. Free-hand sketch recognition had not attracted much attention until very recently due to the lack of touch sensitive devices.

Seddati and et.al proposed a ConvNet that outperformed state-of-the-art results in the TU-Berlin sketch benchmark in 2016 [11]. In this paper, they presented a dual system for sketch classification and similarity search. They used a variant of AlexNet [8]. It was the third attempt of using ConvNets for handsketch recognition. Before that, researchers used to create hand crafted features for classifying the sketch images. One of the interesting thing Seddati and et. al did was that they reversed the pixel value of the image. They replaced all white pixels with black pixels and black pixels with white pixels.

Yang and et.al also proposed SketchANet architecture (CNN based architecture) which produced an accuracy of 74.9 % and used multi-scale multi-channel deep neural network framework [13]. They used a architecture that is very similar to the convolutional architectures used in photo oriented CNNs. The unique aspects of their architecture were larger first layer filters, high dropout,larger pooling size and no local response normalization.

In 2017, Google published the largest collection of black

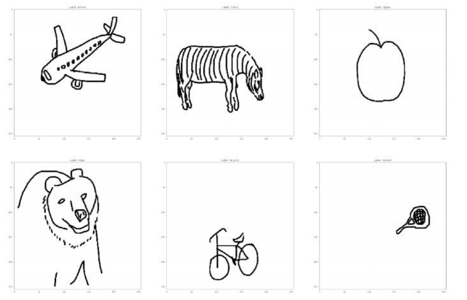

Figure 1. Subset of real images of different classes from the training sets

and white drawing of 345 categories [3]. It contains 50 million drawing contributed by the players of the game Quick, Draw!.

Sketch Classification is a fine grain classification task due to presence of low inter class variation and high intra class variation.A lot of work on fine grain classification has been using transfer learning. ImageNet is such a large dataset and severaldeep architectures have already been trained on it, they are a good source of pre-trained models for fine grain classification [9].

3. Dataset And Preprocessing

We used the publicly available Sketchy database, the first large-scale collection of sketch-photo pairs [10]. It contains 20000 sketches objects across 125 categories. The sketch distribution across 125 categories seems to be balanced.

We used a train-dev-test split of 0.9-0.05-0.05 for dividing our dataset. Our dataset contains 20,000 sketches of 125 categories. There were 18001 training sketches, 999 dev sketches and 1000 test sketches.

The original size of the sketch image was 1100*1100. We scaled down the image from 1100*1100 to 224*224. This is a tradeoff we made due to the limited computation power and limited time for this project. Next, we followed the footsteps of Sedati and et al [11]. where they flip the black and white pixels. The reason behind the flipping the pixel is because the original image is sparse. It contains majority of white pixels. The example of this is given in Appendix 15.

The subset of images taken from the training set is shown in Figure 1. Diving more into the dataset, we definitely see that the dataset possesses high intra class variation and low inter class variation. For example, in Figure 2, we can see how much variation is in the shape of chicken.

Because of low number of training images, we decided to use image augumentation to augment the dataset. We used techniques such as rotation, width shift, height shift, angle shift, zooming, flipping to augment the dataset for better accuracy in this classification task.

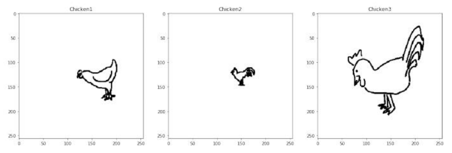

Figure 2. High variation with the chicken class

We also analysed the distribution of labels across the dataset. The distribution of the labels/classes across the dataset was balanced. We did this analysis because the distribution of labels is very influential in choosing the appropriate metric for assessing the performance of our model.

4. Methods

4.1. Framework and Hardware

We used Keras which is a deep learning framework which runs on the top of tensorflow and written in Python [2]. We chose Keras because it is very easy to prototype and experiment different models in Keras. This feature is really essential in a school project like CS230. We trained all models on AWS DeepLearning AMI which included 2 NVIDIA K80 GPUs.

4.2. Metrics/Loss Selection

We selected following loss functions and quantitative and qualitative metrics for evaluating our models.

4.2.1 Loss Function Selection

Since the problem is classification problem and there are 125 categories, we will be using cross entropy loss as the loss function. The cross entropy loss function is

$$\mathcal{L}(X,Y) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \ln a(x^{(i)}))$$

where a(x) is the output of the model given input x.

4.2.2 Qualitative

We selected class activation maps and the image of the activation layer for conducting quantitative analysis. A class activation map for a particular class indicates the selected image regions used by the CNN models to identify that category. Image of the activation layer per channel is an image of a channel that is the result of the convolution of the image. For example:in our baseline model, our first convolutional layer generates an dimension of 112 * 122 * 16. There are 16 channels in this output. If we take one channel, we can get 16 images of dimension 112 * 112 * 112. The nature of images in the earlier layers helps in edge detection and in the later layers they engrave complex features.

4.2.3 Quantitative

We will be using top 1 accuracy, top 5 accuracy, precision, recall and f1 score for doing the quantitaive analysis. Precision measure of all the total images in the dataset how many we classified as that class. Recall measures of all the total true images of a class how many we classified as of that class. F1 score is the harmonic mean of recall and precision.

Top-5 accuracy is the accuracy a model achives when a model predicts the five classes and the correct class is present in any of those five classes. Top-5 accuracy will be higher than top-1 accuracy. We included top 5 accuracy because we have low inter class variation.

4.3. Baseline Model

It contains sequence of conv-max layers followed by fully connected layers at the end. It includes four convolutional layer followed by downsampling operations. After the four covolutional layer, we use two fully connected layers. The first fully connected layer use relu as activation and second fully connected layer(output layer) use softmax as the activation.We introduced dropout in the two fully connected layers. The baseline model was trained for 80 epochs using multi-class cross entropy loss function and rms optimizer [12]. The detailed structure of the model is given in the appendix section

4.4. Advanced Models

We chose ResNet and DenseNet as two advanced models for this task. All prior research work either did not use any neural network or used AlexNet. No one has experimented the task of sketch classification either using ResNet and DenseNet. Due to limited training data and lack of enough computational resources, we will be finetuning the pretrained ResNet and DenseNet on ImageNet.

4.4.1 Residual Neural Networks

Deeper neural networks are more difficult to train. We suffer from vanishing and exploring gradients which hinder the flow of gradient from output layer to the intial layers. To solve this problem, He et al. presented residual learning framework which include layers that learn residual functions with the reference to the layer inputs, instead of learning unreferenced functions [4]. ResNet also uses batch normalization layer and uses convolutional layer instead of fully connected layers at the end of the network. ResNet won almost every ImageNet competition including ILSVRC (ImageNet Large Scale Visual Recognition Challenge) classification competition in 2015. The originally proposed ResNet consisted of 152 layers. ResNet is only composed of small sized-filters of 1x1 and 3x3 with the exception of first convolutional layer of kernel size 7 * 7.

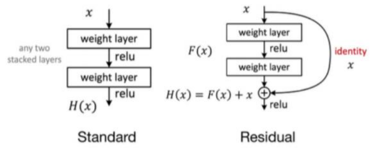

Figure 3. ResNet Connection vs Standard Connection

Every single layer is followed by batch normalization and ReLU activation. Due to lack of extensive computation resources, we used the truncated version of ResNet of 50 layers. The difference between a residual layer and standard layer connection is shown in Figure 3.

Model	Parameters
Baseline	231,717
ResNet50	23,843,837
DenseNet121	7,165,629

Table 1. Total Parameters of Different Models

We freezed the initial layers and trained the layers that were closed to the output layer.

4.4.2 DenseNet

As ResNet showed that you can increase the number of layers without decreasing the accuracy, Huang et al. introduced Dense Convolutional Network called as DenseNet where each layer connect to every other layer in a feed-forward fashion [5]. In traditional convolutional networks with L layers, we have L connection whereas in DenseNet we have $\frac{L*(L+1)}{2}$ connections. Here is the example of a DenseNet taken from the original paper itself in Figure 13 in Appendix section.

4.5. Hyper Parameter Tuning

We spent a lot of time tuning the hyper parameter for three models. The main parameter we tuned were optimizer, learning rate, learning decay rate, number of epochs to train and the batch size.

Generally, batch size is dependent upon the memory of GPU and the number of GPUs. We used 32 as the batch size for ResNet and DenseNet when we were using 1 K80 GPU whereas we used 64 as the batch size for both when we were using 2 K80 GPUs. We used 128 as the batch size for our baseline model.

After batch size, we experimented with the optimizer. We kept the learning rate and learning rate decay constant and experimented with three popular optimizers namely Adam, RMS Prop and SGD. The graph for the loss for three of the optimizers is given in Figure 5.

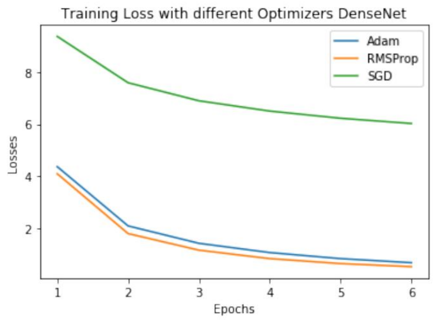

Figure 4. Optimizers vs Loss Graph

We trained each of the models for 6 epochs and selected the best optimizer by seeing the loss chart. For example: we selected RMSProp for DenseNet and Adam for baseline and ResNet. SGD performed worst in all experiments.

After finalizing the optimizer, we experimented with the learning rate in the similar fashion. We will set the learning rate decay constant and use the best optimizer over different learning rates. After that, we will choose the learning rate that gives the lowest loss. For example: we selected 0.0001 as the learning rate for DenseNet as the loss was lowest when we used that learning rate.

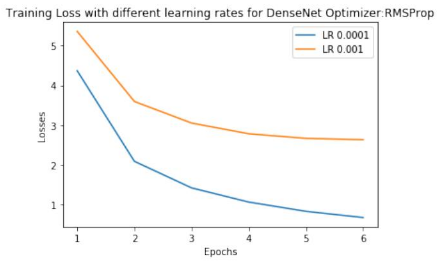

Figure 5. Learning Rate vs Loss Graph

We applied the same procedure for tuning the best learning rate and the number of epochs.

4.6. Training Curves

Here are the results of training curves for all models. The curves of training loss and dev loss for naive baseline CNN, ResNet50 and DenseNet50 are shown in Figure 6, Figure 7 and Figure 8 respectively.

4.7. Result

4.7.1 Quantitative

The accuracy obtained from all of the three models for the test set is described in the Table 2.

Also, here is another table 3 showing the average mean precision, accuracy and F1 score of all three models.

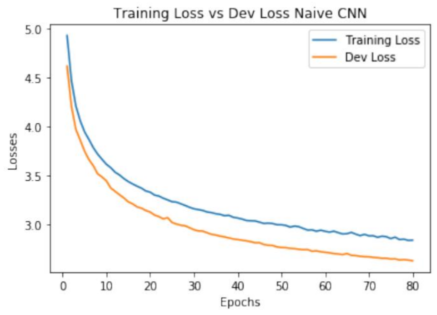

Figure 6. Naive CNN Loss

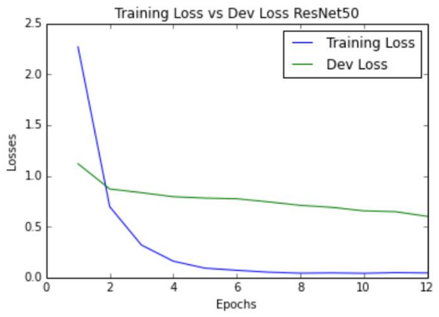

Figure 7. ResNet50 Loss

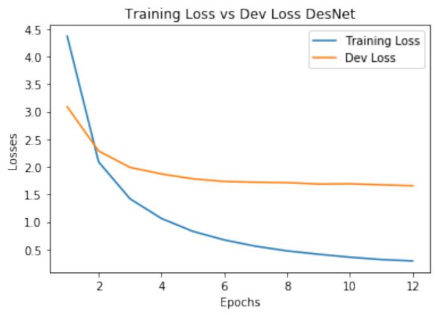

Figure 8. DenseNet Loss

Model	Top 1	Top5
Baseline	35.40%	63.8%
ResNet50	82.20%	94.6%
DenseNet121	77.03%	87.32%

Table 2. Top1 and Top5 accuracy of three models

Let us see the confusion matrix[1] built across 133 classes for baseline CNN in Figure 9 for ResNet and in Figure 10 for DenseNet. The increase in intensity red dots significes the increase in misclassification.

From the above figures, we can observe that the intensity

[1]Source code for confusion matrices at https://github.com/mljs/confusion-matrix

Model	Precision	Recall	F1 Score
Baseline	35.79%	35.40%	32.85%
ResNet50	83.72%	82.20%	82.00%
DenseNet121	77.32%	77.03%	75.53%

Table 3. Precision and Recall of DenseNet, ResNet and Baseline Model

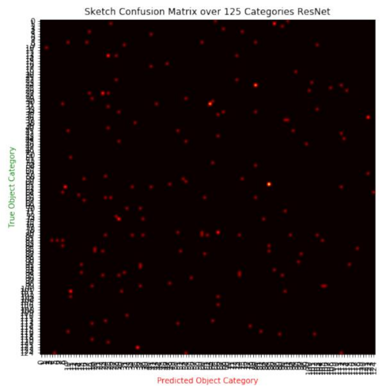

Figure 9. Confusion Matrix for ResNet50

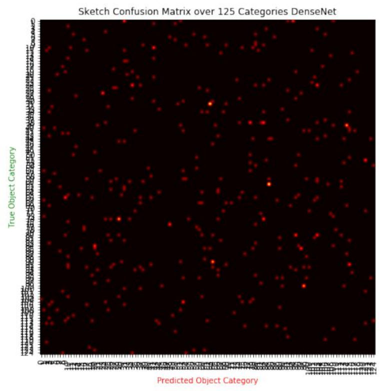

Figure 10. Confusion Matrix for DenseNet121

of red dots is lower in ResNet50 compared to DenseNet121 which supports our result.

4.7.2 Qualitative

We used class activation map for conducting the qualitative analysis. CAMs helps us to find the discriminative regions in image that made the model to classify the image into a specific type.We constructed the CAMs by taking the gradients of the last convolutional layer with respect to the input image, normaliing the output and superimposing the output over original image.

As we can see from Figure 12, the ResNet model focused on the middle part of the image where the aeroplane was. It ignored the outer surrounding completely and we can also

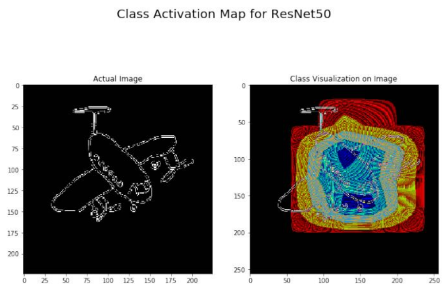

Figure 11. ResNet Class Activation Map for Aeroplane

see the level of focus on the body of the aeroplane. The background was ignored.

Here is the CAM from the DenseNet121 model on the image of aeroplane.

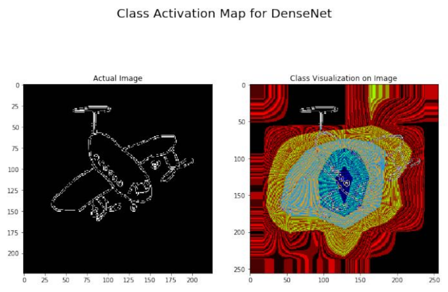

Figure 12. DenseNet Class Activation Map for Aeroplane

As we can see that the focus is concentrated in the middle, but still the focus is not as dense as ResNet model. Even though the focus is not that concentrated, it still predicted the right class. This also supports the fact that the accuracy of ResNet was little better than the DenseNet. We have also shown how CNNs extract features by showing the channels of activations of first Convolutional layers in Appendix 17, 16, 18 and 20.

5. Conclusion

In conclusion, ResNet50 beats DenseNet121 in terms of accuracy and performs better. Also, we see the power of transfer learning. The baseline CNN which was trained from scratch showed an accuracy of 20% which is very low. In contrast, it is very easy to overfit the ResNet model as we can see that the gap between training loss and dev loss is bigger in ResNet50 model as compared to DesNet model.

We also tried to compare/benchmark the performance of ResNet50 and DenseNet121. As a result, we kept the number of epochs same for both ResNet50 and DenseNet121. ResNet50 has better precision and accuracy. We also saw the power of the fine tuning and transfer learning. Our baseline CNN was very similar to SqueezeNet which is equally good model for CNN [6]. However, due to lack of large training data, our baseline model was really bad at making the prediction. Also, it did not have that capacity as compared to deep DenseNet and ResNet.

6. Future Work

We downsampled the image from 1100*1100 to 224*224 to match the size of ResNet and DenseNet as well to decrease the computational load. For the future, we would like to not downsample the image because downsampling the image can cause Convnets to ignore some ignore features responsible for the classification.

Next, in tasks like sketch classification, image segmentation and image resizing are very common. We would like to run our model on multiple sections of the image and use the majority voting to get the predicted class. Also, we should be using the heatmap intensity of the class activation map to get the weighted majority voting. This helps in ignoring the background information which might account into different prediction class.

7. Acknowledgement

We would like to thank CS230N staffs and course assistants for the great quarter. We would like to thanks Amazon Web Services for providing us with AWS Cloud Credits running experiments.

We would like to thank Franois Chollet for this code on finding the class activation map on Keras [1].

References

[1] F. Chollet. Class activation maps. `https://github.com/fchollet/deep-learning-with-python-notebooks`, 2015.

[2] F. Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[3] Google. Quick draw dataset. `https://github.com/googlecreativelab/quickdraw-dataset`, 2017.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[5] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[6] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$1mb model size. *CoRR*, abs/1602.07360, 2016.

[7] M. F. A. Jabal, M. S. M. Rahim, N. Z. S. Othman, and Z. Jupri. A comparative study on extraction and recognition method of cad data from cad drawings. In *2009 International Conference on Information Management and Engineering*, pages 709–713, April 2009.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[10] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 2016.

[11] O. Seddati, S. Dupont, and S. Mahmoudi. Deepsketch: Deep convolutional neural networks for sketch recognition and similarity search. In *2015 13th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6, June 2015.

[12] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[13] Y. Yang and T. M. Hospedales. Deep neural networks for sketch recognition. *CoRR*, abs/1501.07873, 2015.

8. Appendix

Layer	Dimension
Conv	224 * 224 * 16
Max Pool	112 * 112 * 16
Conv	112 * 112 * 32
Max Pool	56 * 56 * 32
Conv	56 * 56 * 64
Max Pool	28 * 28 * 64
Conv	28 * 28 * 128
Max Pool	1 * 1 * 128
FC1	1 * 1 * 512
Dropout	
RELU	
FC2	1 * 1 *133
Dropout	
Softmax	

Table 4. Baseline Model Detailed Structure

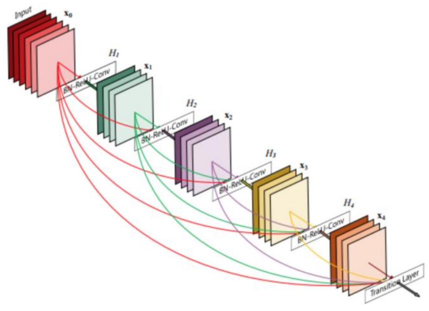

Figure 13. DenseNet Structure

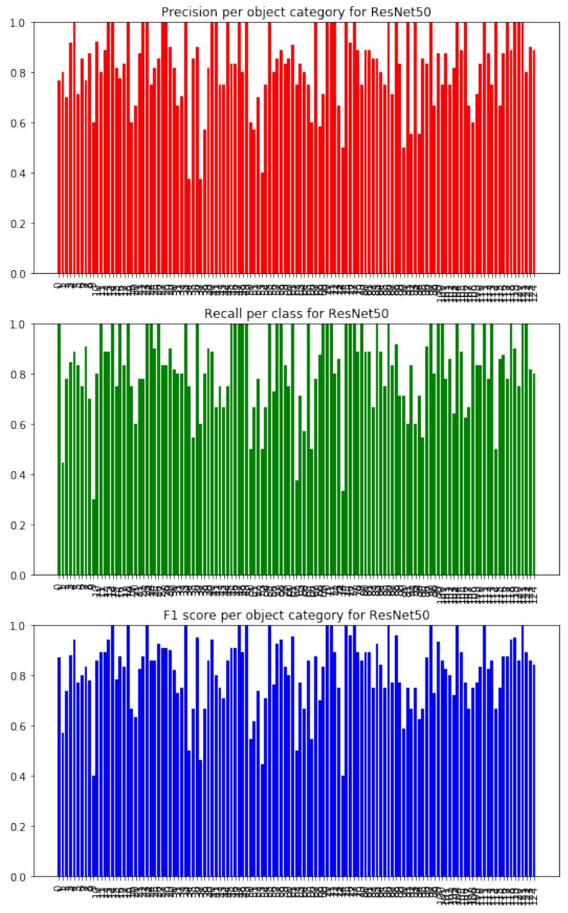

Figure 14. Precision,Recall and F1 of ResNet

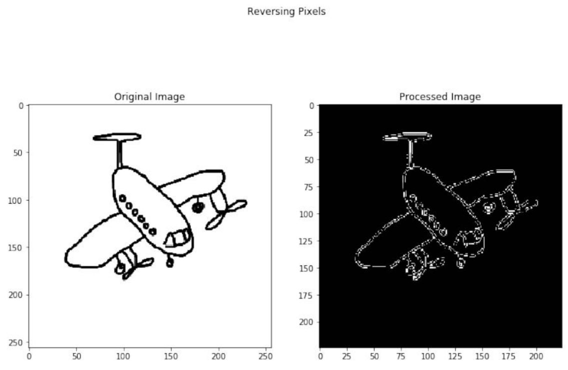

Figure 15. Reversing Pixels

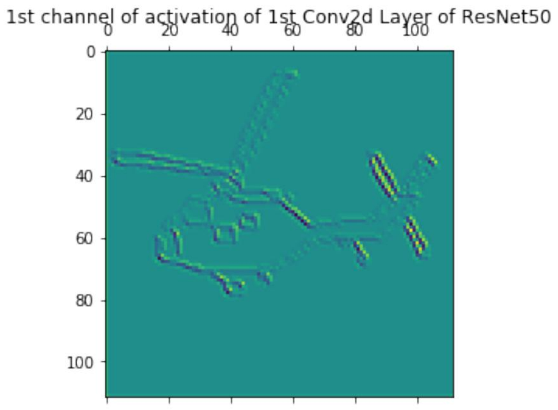

Figure 16. 1st Channel of 1st Activation Layer ResNet for Aeroplane

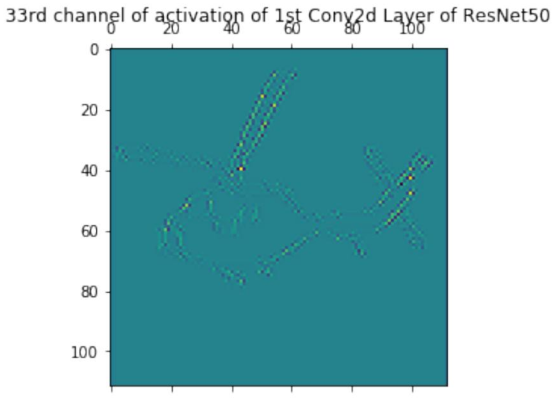

Figure 17. 33rd Channel of 1st Activation Layer ResNet for Aeroplane

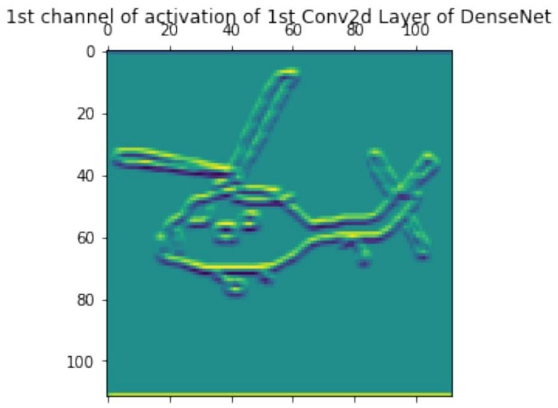

Figure 18. 1st Channel of 1st Activation Layer DenseNet for Aeroplane

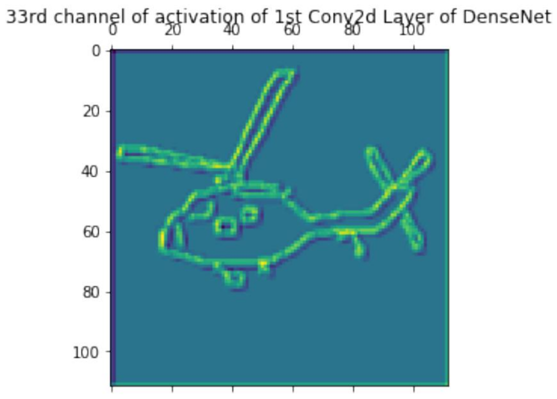

Figure 19. 33rd Channel of 1st Activation Layer DenseNet for Aeroplane

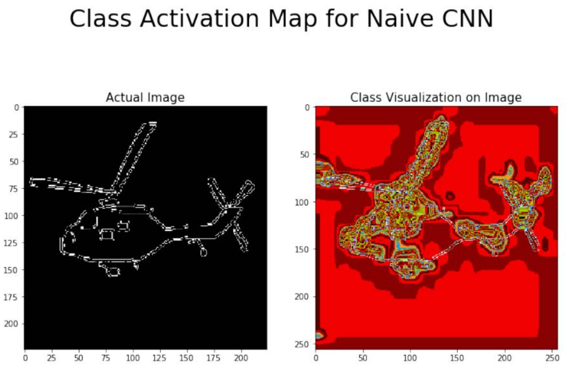

Figure 20. Class Activation Map for Naive CNN