Analysis of Machine Learning models trained on images of drawings

Team name- Pathfinder
Team members- Balaji Balasubramanian (20182628), Niranjan Niranjan (20182630)

1. Introduction

Image Classification is a fundamental problem in Machine Learning. In this competition, we have black and white hand drawings of 6 categories taken from the Quickdraw dataset. Quick, Draw! is a game in which the players draw a picture of an object and a neural network model guesses what the drawings represent. There are 1500 images in the training dataset, 250 from each category and there are 60000 images in the test dataset.
In this paper, we cover data preprocessing steps like data normalization and data augmentation, various machine learning algorithms like Convolutional Neural Network, Support Vector Machines[4] and techniques like hyperparameter tuning, train-validation split and ensembling methods to improve their performance.

2. Feature Design

2.1 Preprocessing

The various data preprocessing steps are:-

- Data normalization- The image array has values ranging from 0 to 255. We divide the image array by 255 to convert the values from 0 to 1. This makes it easier for the Machine Learning model like Convolutional Neural Network and Support Vector Machines to learn from the data.
- Converting the labels to one-hot encoding before passing it to the Convolutional Neural Network model.
- Resizing the image to 28 x 28 x 1 dimension before passing it to the Convolutional Neural Network model.

2.2 Data Augmentation

The training data has only 1500 images which are insufficient to train Deep Learning models like Convolutional Neural Networks. Data Augmentation techniques are used to artificially create new training data. These are most commonly used to train images and can reduce the variance of the model. We found that Data Augmentation improved the accuracy of our CNN model by 15% on the public leaderboard. We used Keras[1] Image Data Generation to perform data augmentation.
The various data augmentation techniques we tried are:-

- Width and height shift- This shifts the image horizontally and vertically. We found that 0.15 is an ideal value for the width and height and width shift i.e. the image is by 15% of its width and height.
- Horizontal and vertical flip- We found that horizontal flip increased the accuracy of the model while vertical flip decreased the accuracy of the model. Hence, we used only horizontal flip.
- Rotation and shear - Both rotation and shearing are performed for 15 degrees.
- Zoom- We performed data augmentation method ‘zoom’ to zoom the image by 0.15 of the image size.

3. Algorithms

The various algorithms used on the dataset are:-

I. Convolutional Neural Network[5]

The Convolutional Neural Network model was implemented using Keras[1] library. The architecture of the model is given below. It is inspired by the architecture of VGG Net[7].

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer_conv1 (Conv2D)         (None, 28, 28, 32)        320
_________________________________________________________________
layer_conv2 (Conv2D)         (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
layer_conv3 (Conv2D)         (None, 14, 14, 64)        18496
_________________________________________________________________
layer_conv4 (Conv2D)         (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
layer_conv5 (Conv2D)         (None, 7, 7, 128)         73856
_________________________________________________________________
layer_conv6 (Conv2D)         (None, 7, 7, 128)         147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 1000)              1153000
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 6006
=================================================================
Total params: 1,445,438
Trainable params: 1,445,438
Non-trainable params: 0
_________________________________________________________________
```

The first two layers are convolutional layers with 32 neurons each with Rectified Linear Units(ReLU) activation function, and the third layer is a max pooling layer. The next two layers are convolutional layers with 64 neurons each with ReLU activation, and then there is a max pooling layer. The two layers after that are convolutional layers with 128 neurons each with ReLU activation, and then there is a max pooling layer. The next layer is the flatten layer which converts (3,3,128) dimensional output from the convolutional layer to 1152 dimensional output. Then there is a dense layer of 1000 neurons with ReLU activation. The final layer has 6 neurons that correspond to the 6 output categories and it has softmax activation.

The number of neurons in the convolutional layers are more in the later layers than the initial layers because the later layers need to learn a more complex representation than the initial layers.

Nadam optimizer[2], categorical cross-entropy loss function, and a batch size of 32 was used. This model was trained for 120 epochs.

II. Logistic Regression[6]

Logistic Regression is a simple machine learning model and it performs well in simple datasets with few training samples. We implement Multinomial Logistic Regression with L2 regularization (hyperparameter 'C' in Sklearn) with Scikit-Learn library. In Scikit-Learn terms, the hyperparameter 'C' is inversely proportional to regularization strength i.e. if the value of C is low, the model is simple and if the value of C is high then the model is more complex.
In this competition, the number of training samples is less but the samples are complex images and the model performed worse than CNN. The highest validation accuracy was achieved when the value of C was equal to 0.01.

III. Support Vector Machines(SVM)[4]

Support Vector Machines (Support Vector Classifier in Scikit-Learn terms) is the third model tried. It uses Kernel Trick to deal with non-linear data and uses maximum marginal hyperplane while separating the classes to improve the generalization. Both Rbf and Polynomial kernel was tried and it was found that Polynomial Kernel with degree 3 performed the best. Similar to Logistic Regression, L2 Regularization was used. In Scikit-Learn terms, the hyperparameter 'C' is inversely proportional to regularization strength. The highest validation accuracy was achieved when the value of C was equal to 1.3.
SVM performed better than Logistic Regression but worse than linear regression. It performed better than logistic regression due to kernel trick and maximum marginal hyperplane.

4. Methodology

Various techniques were used such as Train-Validation Split, K-Fold Cross Validation, Hyperparameter Tuning, Regularization and Ensembling.

I. Train-Validation Split

The training data was split into train and validation sets. The train set had 90% of the data(1350 samples) and the validation set had 10% of the data(150 samples) while training the Convolutional Neural Network. Train-Validation Split has a bias due to the selection of validation set but it is much faster to run than K-Fold Cross-Validation. Hence, it was used a lot while training the Convolutional Neural Network because the model takes a long time to train and Train-Validation Split is much faster than K-Fold Cross Validation.
While training Support Vector Machines, Logistic Regression and Random Forest, Train set had 80% of the data(1200 samples) and the validation set had 20% of the data(300 samples)

II. K-Fold Cross-Validation

K-fold Cross Validation removes the selection bias caused by manual selection of the validation set. We set the value of k as 5. Hence, 4 folds were used as the training set, and 1 fold was used as the validation set. Five iterations were done and each iteration had a different validation set. K-Fold Cross Validation was used to test a few hyperparameter values that had performed well on Train-Validation Split to get the best hyperparameters. K-fold Cross Validation was used for the hyperparameter tuning of all the four models.

III. Hyperparameter tuning

The various hyperparameters tuned while training the Convolutional Neural Network model are:-

- The number of convolutional and max pooling layers. We tried varying the range of convolutional layers from 1 to 8.
- The number of neurons, kernel size and strides in each convolutional layer. We tried varying the number of neurons from the range of 16 to 256, kernel size of 3,5,7 and stride of 1,2.
- Number of neurons in the dense layer
- Selecting the optimizer. We tried Nadam[2] and Adam[3].
- Selecting the learning rate. We tried varying the learning rate from 0.01 to 0.0001.
- We tried batch sizes of 32,64 and 128
- We varied the number of epochs from the range of 50 to 250.

For the Support Vector Machines, RBF kernel and Polynomial Kernel were tried and it was found that Polynomial Kernel of degree 3 works better. The value of hyperparameter C (L2 Regularizer) varied from the range of 0.1 to 3.
For Logistic Regression, the value of hyperparameter C(L2 Regularizer) was varied from the range of 0.001 to 0.6.

IV. Regularization

L2 Regularizer was used for Support Vector Machines and Logistic Regression and it worked well.
Dropout was tried while training Convolutional Neural Network but it reduced the model performance. Hence, it was removed.

V. Ensemble Methods

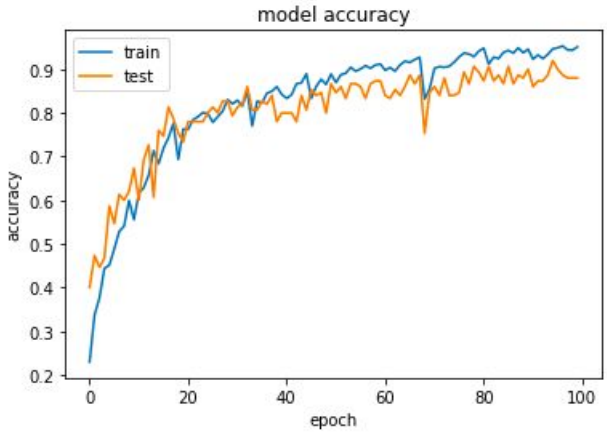
For improving the accuracy, an ensemble of 4 CNN models was made. Here, 4 CNN models with slightly different configurations, data augmentation, and structure(number of layers and neurons in each layer) were made and their predictions were combined, in which more weightage was given to the model with higher accuracy and less weightage was given to the model with lower accuracy.
Ensembling increased the accuracy of the model by 2%.

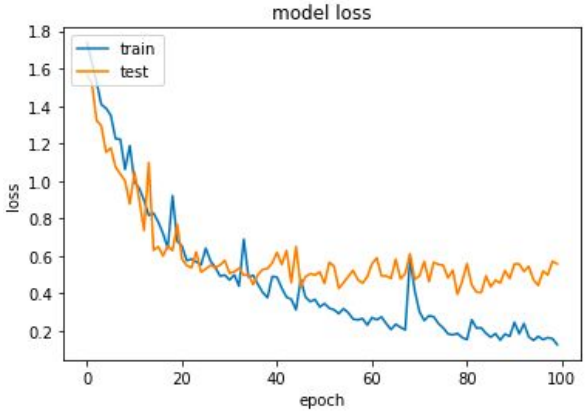
5. Results

Now we discuss the results of 3 models:-

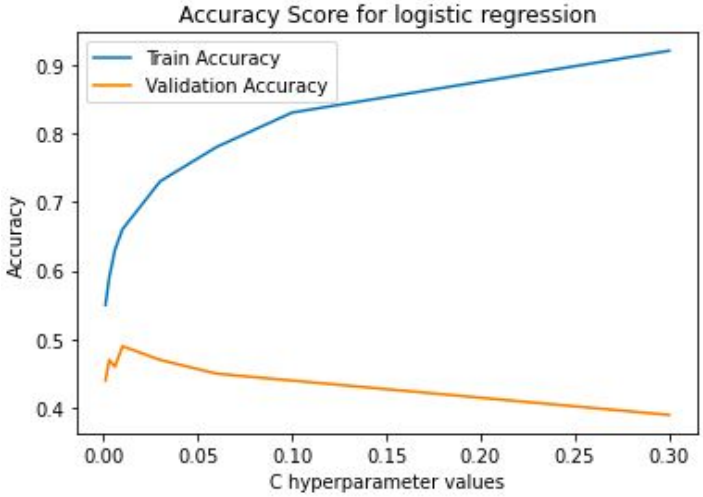
I. Convolutional Neural Network

The graph below shows the change of model loss and model accuracy over 100 iterations for the Convolutional Neural Network model. The structure of the model is the same as that given in the Algorithms section (6 Convolutional Layers). Train(0.9) and validation(0.1) split is used to calculate the scores.





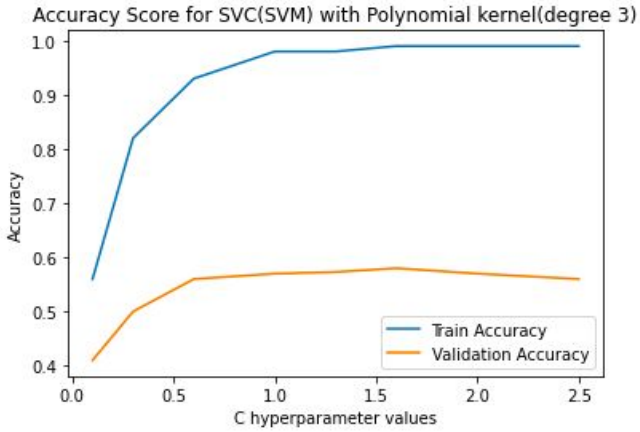
II. Logistic Regression



For Logistic Regression, the regularizer used is the L2 regularizer which is controlled by Scikit-Learn's C hyperparameter. C is inversely proportional to regularization strength.
The graph above shows the model's Train and Validation Accuracy varies with the change in C. Train(0.8) and validation(0.2) split is used to calculate the scores. The highest validation accuracy is achieved when the value of C is equal to 0.01.

III. Support Vector Machines(SVM)

SVM with a polynomial kernel of degree 3 was used because it performed better than RBF kernel. Similar to Logistic Regression, L2 regularizer was used which is controlled by Scikit-Learn's C hyperparameter. The graph below shows the model's Train and Validation Accuracy varies with change in C. Train(0.8) and validation(0.2) split is used to calculate the scores. The highest validation accuracy is achieved when the value of C is equal to 1.6.



As we can see from the above 4 graphs, Convolutional Neural Network performs the best, then comes Support Vector Machines and Logistic Regression performs the worst. The best model on Kaggle Leaderboard was an ensemble of 4 CNN models and it scored 0.87222 on the public leaderboard and 0.87607 on the private leaderboard.

6. Discussion

The Convolutional Neural Network model managed to cross the Random Prediction Baseline and TA Baseline and worked well for this task. Some of the ideas that can be tried are transfer learning to build a better representation, cyclical learning rate to improve optimization, better hyperparameter tuning using techniques like early stopping and trying recent research ideas like Capsule Network and Self Supervised Learning.

7. Statement of Contribution

The problem definition, developing the methodology, data analysis and report writing was done by both Balaji and Niranjan
Balaji built the initial Convolutional Neural Network model with data augmentation using Keras. After that, Balaji and Niranjan collaborated on hyperparameter tuning (and its coding) to improve the performance of the model by changing the number of layers and neurons in each layer, changing the learning rate, data augmentation methods and the number of iterations.
Balaji and Niranjan also collaborated on building the machine learning models Logistic Regression and Support Vector Machines and tuning its hyperparameters.
We hereby state that all the work presented in this report is that of the authors.

8. References

1.Tensorflow and Keras- https://www.tensorflow.org/tutorials/images/cnn
2.Nadam Optimizer- http://cs229.stanford.edu/proj2015/054_report.pdf
3.Adam Optimizer- https://arxiv.org/abs/1412.6980
4.Sklearn Support Vector Machines- https://scikit-learn.org/stable/modules/svm.html
5.Deep Learning book- https://www.deeplearningbook.org/
6. Sklearn Logistic Regression- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
7. VGG Net- https://arxiv.org/pdf/1409.1556.pdf