Project Report

on

Image Classification of 10 Knots using Convolutional Neural Networks

By,

Group Name: Neural_Masters

Varun Sharma	varun15sharma15@csu.fullerton.edu
Vibhav Kapadia	vibhav@csu.fullerton.edu
Milan Ghori	ghori18@gmail.com
Jithin Eapen	jithin.john@csu.fullerton.edu

Instructor,

Venkatramanan Krishnamani

Submitted on - 05/14/2018

For course - CPSC 585 Artificial Neural Networks

1. Task

We worked on a classification problem for classifying 10 different types of Knots.

2. Dataset

Original dataset consists of 10 classes of different knots, with 144 images in each class.

3. Technologies used

- Keras – Library for defining, compiling and training neural network.
- Google Colaboratory – Free Online GPU environment with Jupyter notebook.
- Matplotlib – Library for plotting charts.
- Augmentor – Library to create more images.
- FastStone Image Resizer - Tool used to scale down images.

4. Process followed

Resnet

We used a Resnet model originally trained for Cat-Dog classifier. But, the accuracy of the model was only around 50%. First, we tried only 2 classes. But model was predicting only for 1 class.

```
Epoch 1/5
12/12 [==============================] - 1085s 90s/step - loss: 0.7075 - acc: 0.5824 - val_loss: 0.8541 - val_acc: 0.5000
Epoch 2/5
12/12 [==============================] - 1099s 92s/step - loss: 0.4485 - acc: 0.8066 - val_loss: 0.8749 - val_acc: 0.5000
Epoch 3/5
12/12 [==============================] - 1061s 88s/step - loss: 0.3715 - acc: 0.8385 - val_loss: 0.8816 - val_acc: 0.5000
Epoch 4/5
12/12 [==============================] - 1000s 83s/step - loss: 0.3448 - acc: 0.8754 - val_loss: 0.7848 - val_acc: 0.5000
Epoch 5/5
12/12 [==============================] - 967s 81s/step - loss: 0.2731 - acc: 0.8758 - val_loss: 0.8407 - val_acc: 0.5000

<keras.callbacks.History at 0x7fd85076dcd0>
```

Fig: Training and Validation accuracy for Resnet

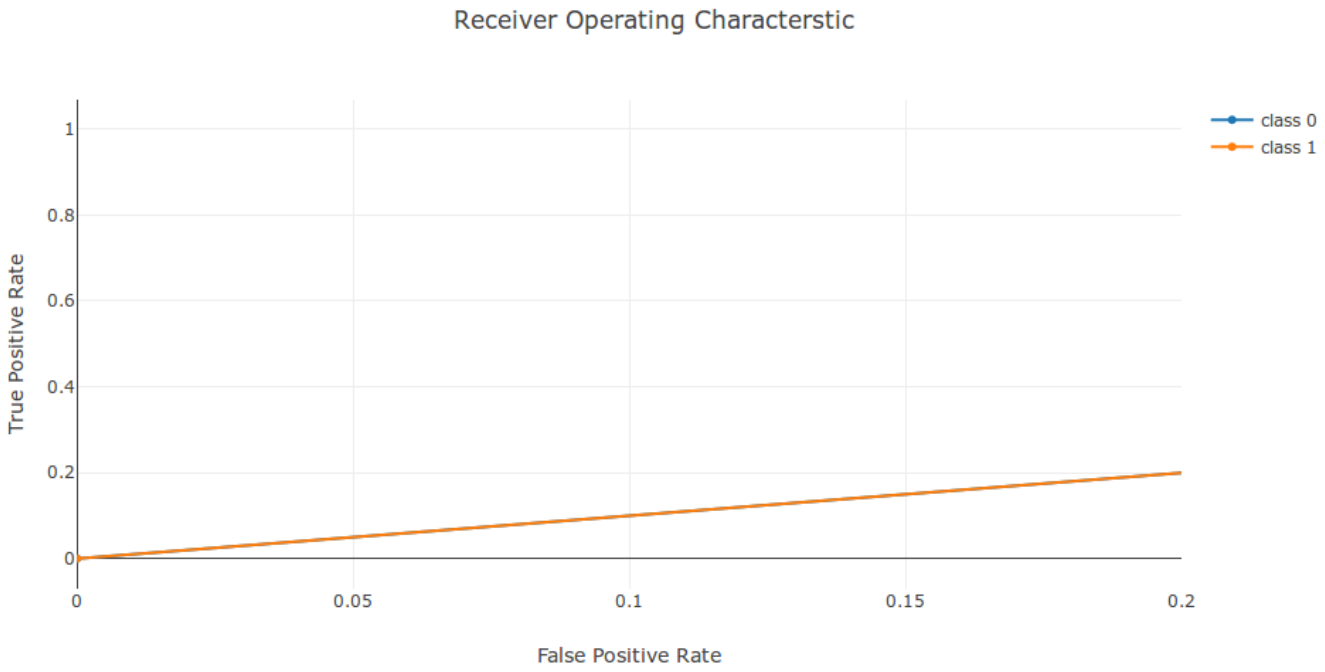


Fig: ROC showing only 1 class

Smaller images

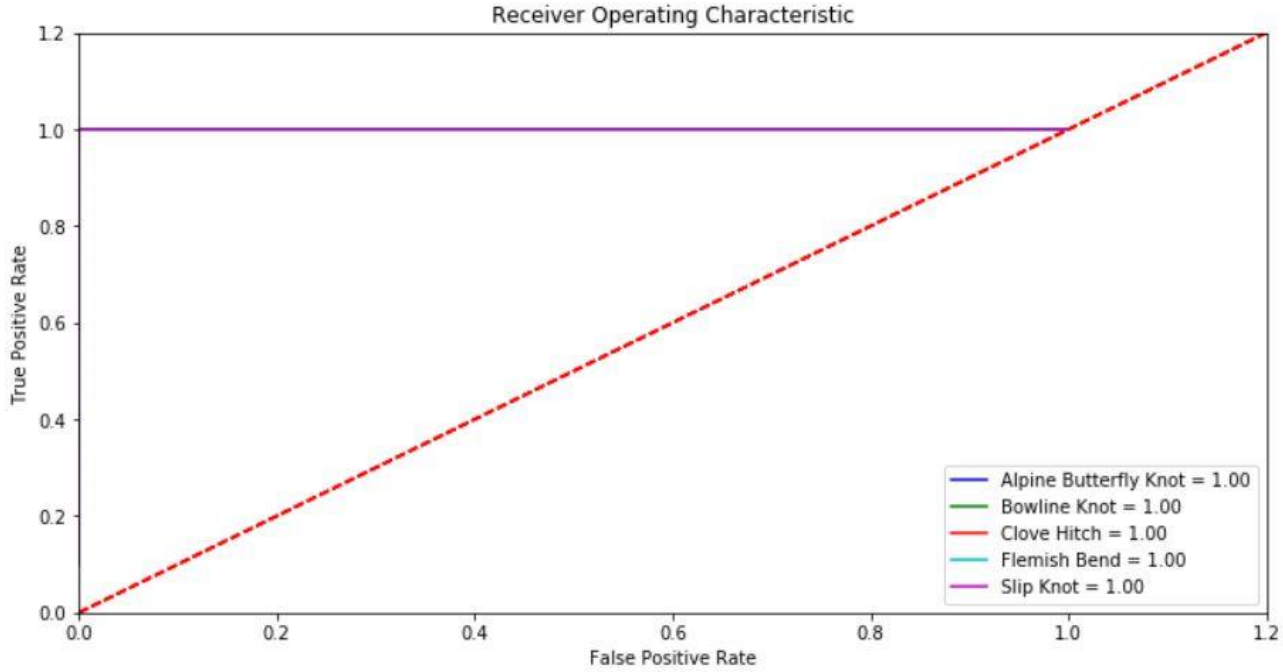
As the original images were too large in size (5000 x 3500 px and approx 7 MB each), the training time was too much. And more importantly, the model was Learning more background feature instead of the Knots. So we rescaled the image to 150*150 size.

Augmented Images

As the original dataset was very small for an image classification problem, we tried to increase the dataset size by augmenting the images. We used python *Augmentor* library to perform rotate, and translate and create a new dataset with 8080 images for Training, 1860 images for Validation and 200 images for Testing.

Custom Model (5 classes)

Next, we created a custom model with 5 class and 11 layers. Below is the ROC curve for it.



Custom Model (10 classes)

We created a custom network model, with 19 layers, consisting of Convolutional layers of kernel size 3*3 and either 32 or 64 filters, and Activations of relu, and Batch Normalization followed by MaxPooling layers of 2*2.

For the output section, we Flattened the inputs, and used Dense layer, a Relu and Dropout layer, and a Softmax output of 10 units for the 10 classes.

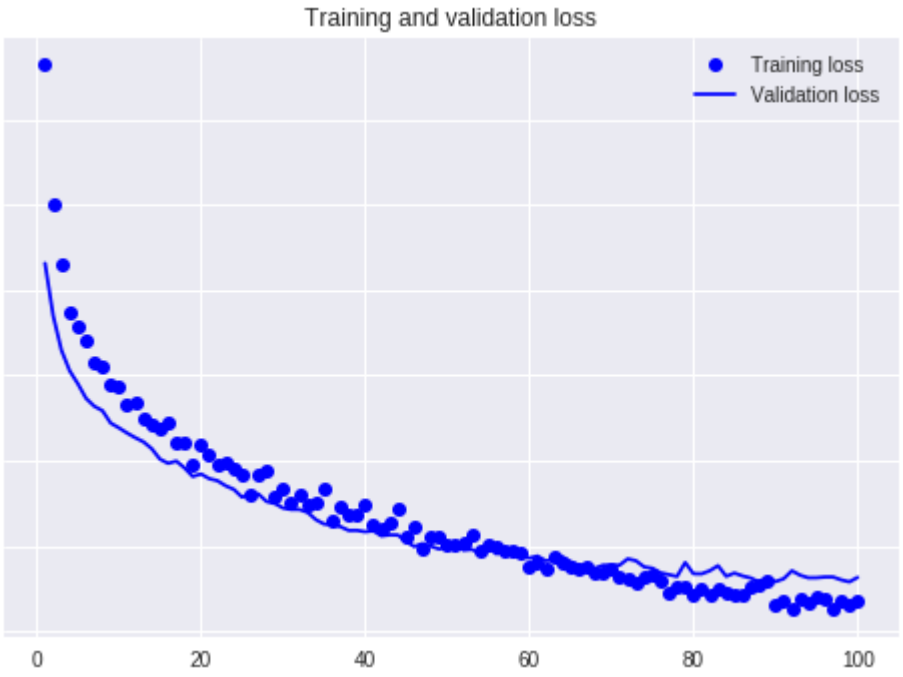
Loss function used was **Categorical_crossentropy** as it is a multi-class problem, and **RMSprop** is used as the type of gradient descent algorithm.

Hyperparameters used

- Learning rate – 0.00001
- Epochs – 100
- Sample per epoch – 808
- Validation steps – 186

Training & Validation

During training & validation, we were able to achieve around 80% accuracy, with very less overfitting (similar Training and Validation curves).



Testing & Prediction

We got a Test accuracy of 87.99% and high AUC for each class, as shown in the ROC curves below.

Fig: ROC curves for 10 classes

Conclusion

We were able to achieve a very high average AUC value of 0.990833