Part 1

Sample 1:

Original

:

Chopped

Sample2:

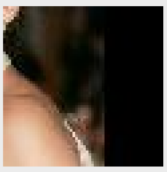
Original:

Cropped:

Sample 3:

Original

Cropped:



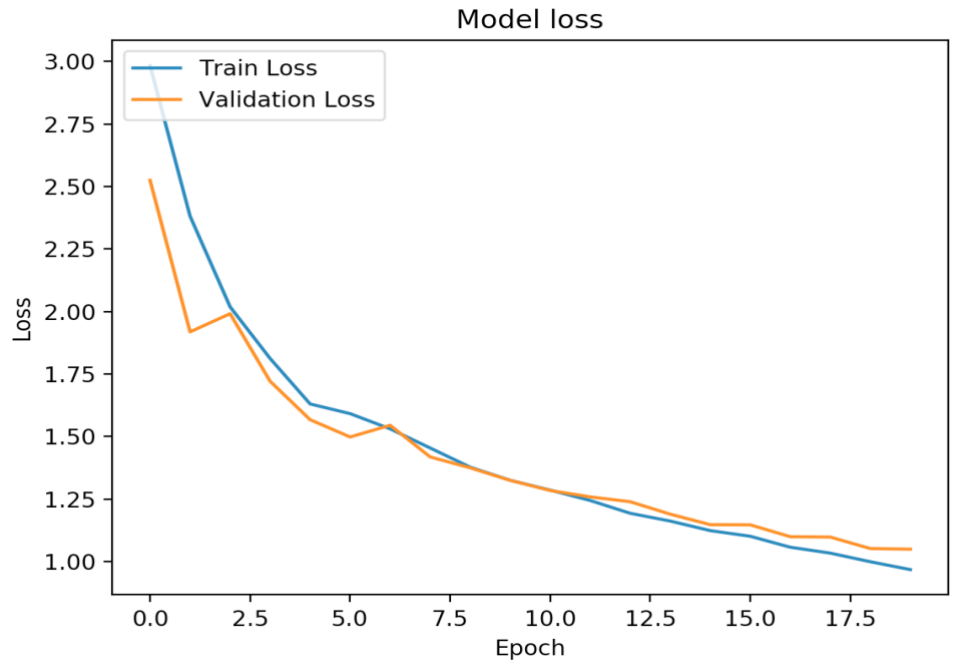
1. The provided data sets are accurate for most bounding box. For example, for image “radcliffe53.jpg” (the first sample above), it successfully cropped character’s face. But for some of the cropped images, they either contain extra “information “or not accurate. For example, for sample 2, it has another person’s face on the right. Also, for sample 3, it doesn’t contain character’s face, so, during the process of data training, it may cause inaccuracy.

2. It is important because without the cropping process, there are a lot of external factor could affect our prediction, like the background, other people’s face, etc. For example, for sample 2, the original image also contains baby’s face, but what we want is the lady’s face as sample during training. By cropping the image, it could prevent the influence of external factors and increase the prediction accuracy.

3. By resizing the image, we could save the compile time without lose an information (because resize an image doesn’t change the amount of data in that image).

Part 2

1.




2. and 3.

In the process of data splitting, the preprocessed image and the image label were splitted into X_train, X_test, y_train, y_test by 0.2 (80% training sets and 20% testing sets). Then for X_train, and y_train, it was further splitted by 0.2 (80% training sets and 20% testing sets) to X_train, X_valid, y_train, y_valid.
Therefore, for X_valid, it contains 20% to the training data.
For y_valid, it contains 20% of the testing data.
For X_train, it contains 80% of the training data.
For X_test, it contains 20% of preprocessed image data.
Preprocessed the input:
For preprocessed image, firstly convert all the images to greyscale, so that all the image only has one channel. Then, use the get data function to get the information from the image and divide the data by 255. After the division, append the image data to preprocessed image list. Therefore, after the iteration, preprocessed image list will have all images ‘data
For label, it uses file name to identify characters. For example, for file name “bracco0.jpg”, since it contains “bracco”, label it with “0”. After go through all the filename, the label list will have all the characters labeled.

| Character Name | Label |
|---|---|
| Bracco | 0 |
| butler | 1 |
| Gilpin | 2 |
| Harmon | 3 |
| Radcliffe | 4 |
| Vartan | 5 |

Activation function used:

1. “relu”: Rectified Linear Unit.
2. “softmax”: Softmax activation function.

Number of hidden layer node selected: 100
Compiling result:

```
2019-12-16 02:41:27.958978: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports
Train on 362 samples, validate on 91 samples
Epoch 1/20
 - 0s - loss: 3.2849 - accuracy: 0.1961 - val_loss: 3.6240 - val_accuracy: 0.1319
Epoch 2/20
 - 0s - loss: 3.0938 - accuracy: 0.1989 - val_loss: 2.3890 - val_accuracy: 0.2418
Epoch 3/20
 - 0s - loss: 1.9970 - accuracy: 0.2597 - val_loss: 1.7592 - val_accuracy: 0.2637
Epoch 4/20
 - 0s - loss: 1.7909 - accuracy: 0.1961 - val_loss: 1.6752 - val_accuracy: 0.3297
Epoch 5/20
 - 0s - loss: 1.6939 - accuracy: 0.2680 - val_loss: 1.7243 - val_accuracy: 0.2857
Epoch 6/20
 - 0s - loss: 1.6824 - accuracy: 0.2680 - val_loss: 1.6299 - val_accuracy: 0.4396
Epoch 7/20
 - 0s - loss: 1.6314 - accuracy: 0.3149 - val_loss: 1.5953 - val_accuracy: 0.3846
Epoch 8/20
 - 0s - loss: 1.6070 - accuracy: 0.4033 - val_loss: 1.5951 - val_accuracy: 0.4835
Epoch 9/20
 - 0s - loss: 1.5756 - accuracy: 0.4530 - val_loss: 1.5909 - val_accuracy: 0.4286
Epoch 10/20
 - 0s - loss: 1.5452 - accuracy: 0.4751 - val_loss: 1.5531 - val_accuracy: 0.4286
Epoch 11/20
 - 0s - loss: 1.5184 - accuracy: 0.5055 - val_loss: 1.5168 - val_accuracy: 0.5385
Epoch 12/20
 - 0s - loss: 1.4874 - accuracy: 0.4779 - val_loss: 1.4923 - val_accuracy: 0.4945
Epoch 13/20
 - 0s - loss: 1.4684 - accuracy: 0.4696 - val_loss: 1.4725 - val_accuracy: 0.5165
Epoch 14/20
 - 0s - loss: 1.4374 - accuracy: 0.5276 - val_loss: 1.4503 - val_accuracy: 0.5604
Epoch 15/20
 - 0s - loss: 1.4178 - accuracy: 0.5442 - val_loss: 1.4339 - val_accuracy: 0.5165
Epoch 16/20
 - 0s - loss: 1.3914 - accuracy: 0.5359 - val_loss: 1.4118 - val_accuracy: 0.5604
Epoch 17/20
 - 0s - loss: 1.3641 - accuracy: 0.5525 - val_loss: 1.3842 - val_accuracy: 0.5495
Epoch 18/20
 - 0s - loss: 1.3444 - accuracy: 0.5663 - val_loss: 1.3682 - val_accuracy: 0.5275
Epoch 19/20
 - 0s - loss: 1.3173 - accuracy: 0.5829 - val_loss: 1.3544 - val_accuracy: 0.5824
Epoch 20/20
 - 0s - loss: 1.2898 - accuracy: 0.5939 - val_loss: 1.3344 - val_accuracy: 0.5604
Test loss: 1.3783191463403535
Test accuracy: 0.48245614767074585
```

## Part 3:

1. For “block4_pool”

```
Train on 362 samples, validate on 91 samples
Epoch 1/20
 - 6s - loss: 8050.5685 - accuracy: 0.2320 - val_loss: 12122.0879 - val_accuracy: 0.5165
Epoch 2/20
 - 3s - loss: 15096.3017 - accuracy: 0.4392 - val_loss: 7251.0254 - val_accuracy: 0.4835
Epoch 3/20
 - 3s - loss: 5104.4824 - accuracy: 0.5470 - val_loss: 1750.1118 - val_accuracy: 0.3956
Epoch 4/20
 - 3s - loss: 1431.9942 - accuracy: 0.5193 - val_loss: 2024.0032 - val_accuracy: 0.2637
Epoch 5/20
 - 3s - loss: 165.4358 - accuracy: 0.8674 - val_loss: 458.3141 - val_accuracy: 0.6923
Epoch 6/20
 - 4s - loss: 198.4392 - accuracy: 0.8729 - val_loss: 636.7608 - val_accuracy: 0.6703
Epoch 7/20
 - 3s - loss: 108.5728 - accuracy: 0.8812 - val_loss: 363.0591 - val_accuracy: 0.7912
Epoch 8/20
 - 3s - loss: 5.9445 - accuracy: 0.9779 - val_loss: 395.1650 - val_accuracy: 0.7912
Epoch 9/20
 - 3s - loss: 2.3046 - accuracy: 0.9917 - val_loss: 601.2181 - val_accuracy: 0.7143
Epoch 10/20
 - 3s - loss: 8.9258 - accuracy: 0.9834 - val_loss: 551.5802 - val_accuracy: 0.7143
Epoch 11/20
 - 3s - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 441.8547 - val_accuracy: 0.7363
Epoch 12/20
 - 3s - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 379.5051 - val_accuracy: 0.7692
Epoch 13/20
 - 3s - loss: 0.2851 - accuracy: 0.9972 - val_loss: 347.0390 - val_accuracy: 0.7912
Epoch 14/20
 - 3s - loss: 1.3411 - accuracy: 0.9972 - val_loss: 334.7331 - val_accuracy: 0.7802
Epoch 15/20
 - 3s - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 333.9252 - val_accuracy: 0.7802
Epoch 16/20
 - 3s - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 333.2542 - val_accuracy: 0.7802
Epoch 17/20
 - 3s - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 332.6701 - val_accuracy: 0.7802
Epoch 18/20
 - 3s - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 332.3909 - val_accuracy: 0.7692
Epoch 19/20
 - 3s - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 332.2582 - val_accuracy: 0.7692
Epoch 20/20
 - 3s - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 332.2023 - val_accuracy: 0.7692
Test loss: 293.2644206264563
Test accuracy: 0.7982456088066101
```

For “block5_pool”

```
Train on 362 samples, validate on 91 samples
Epoch 1/20
 - 0s - loss: 56.6497 - accuracy: 0.2155 - val_loss: 63.4098 - val_accuracy: 0.1648
Epoch 2/20
 - 0s - loss: 34.2746 - accuracy: 0.3425 - val_loss: 35.3666 - val_accuracy: 0.2857
Epoch 3/20
 - 0s - loss: 16.8790 - accuracy: 0.5028 - val_loss: 14.5358 - val_accuracy: 0.4176
Epoch 4/20
 - 0s - loss: 6.0659 - accuracy: 0.6602 - val_loss: 6.4876 - val_accuracy: 0.5604
Epoch 5/20
 - 0s - loss: 2.4501 - accuracy: 0.7348 - val_loss: 4.7478 - val_accuracy: 0.5934
Epoch 6/20
 - 0s - loss: 0.9701 - accuracy: 0.8149 - val_loss: 1.7574 - val_accuracy: 0.7033
Epoch 7/20
 - 0s - loss: 0.3681 - accuracy: 0.8619 - val_loss: 1.7613 - val_accuracy: 0.6374
Epoch 8/20
 - 0s - loss: 0.3139 - accuracy: 0.8702 - val_loss: 1.7064 - val_accuracy: 0.6703
Epoch 9/20
 - 0s - loss: 0.1413 - accuracy: 0.9309 - val_loss: 1.4191 - val_accuracy: 0.7473
Epoch 10/20
 - 0s - loss: 0.1018 - accuracy: 0.9641 - val_loss: 1.3792 - val_accuracy: 0.7363
Epoch 11/20
 - 0s - loss: 0.0567 - accuracy: 0.9779 - val_loss: 1.4165 - val_accuracy: 0.7363
Epoch 12/20
 - 0s - loss: 0.0398 - accuracy: 0.9890 - val_loss: 1.3980 - val_accuracy: 0.7363
Epoch 13/20
 - 0s - loss: 0.0260 - accuracy: 0.9917 - val_loss: 1.3637 - val_accuracy: 0.7143
Epoch 14/20
 - 0s - loss: 0.0204 - accuracy: 0.9917 - val_loss: 1.3571 - val_accuracy: 0.7692
Epoch 15/20
 - 0s - loss: 0.0155 - accuracy: 1.0000 - val_loss: 1.3681 - val_accuracy: 0.7473
Epoch 16/20
 - 0s - loss: 0.0117 - accuracy: 0.9972 - val_loss: 1.3792 - val_accuracy: 0.7473
Epoch 17/20
 - 0s - loss: 0.0097 - accuracy: 0.9972 - val_loss: 1.3934 - val_accuracy: 0.7473
Epoch 18/20
 - 0s - loss: 0.0073 - accuracy: 1.0000 - val_loss: 1.4046 - val_accuracy: 0.7582
Epoch 19/20
 - 0s - loss: 0.0055 - accuracy: 1.0000 - val_loss: 1.4132 - val_accuracy: 0.7582
Epoch 20/20
 - 0s - loss: 0.0044 - accuracy: 1.0000 - val_loss: 1.4180 - val_accuracy: 0.7582
Test loss: 1.3731244185514617
Test accuracy: 0.7719298005104065
```

For VGG16 layer, since “block4_pool” and “block5_pool” has similar accuracy (block4: 0.798, block5: 0.771), but “block5_pool” has significantly lower test loss (block4: 293.26, block5: 1.37). Therefore, I choose “block5_pool” as VGG16 layer.

2. VGG16 yields better accuracy, because for VGG16, we use a CNN to pre-train the image data set. Also, for VGG16, it drives high level features from the image’s visual content using fulling connected layer (from the lecture slide). Also, for VGG16, it has many weight parameters, so it increases the accuracy of image recognition.

Part 4

Original parameter:

Number of Hidden Layer: 1

Number of Node in hidden layer: 100

Dropout: no dropout

Batch size: 128

Epoch: 20

```
Epoch 20/20
 - 0s - loss: 0.8898 - accuracy: 0.7486 - val_loss: 1.0636 - val_accuracy: 0.626
4
Test loss: 1.065111141455801
Test accuracy: 0.6491228342056274
```

Add dropout to be 0.1 and keep other parameters the same:

Number of Hidden Layer: 1

Number of Node in hidden layer: 100

Dropout: 0.1

Batch size: 128

Epoch: 20

```
Test loss: 1.119551840581392
Test accuracy: 0.6228070259094238
```

Base on the output, add dropout doesn't improve the accuracy at lot, it rather slightly decreases the accuracy (0.649-0.622). That is because when the data set is small, adding dropout rate would decrease its accuracy. Adding dropout would only increase the accuracy if the data set is huge, because since the data set is big enough, overfitting is not a problem.

Add another hidden layer and set the node to be 50:

Number of Hidden Layer: 2

Number of Node in first hidden layer: 100

Number of Node in first hidden layer: 50

Dropout: no dropout

Batch size: 128

Epoch: 20

```
Test loss: 1.061182021049031
Test accuracy: 0.6666666865348816
```

Base on the output, add another hidden layer does not improve the accuracy a lot. Because by adding more hidden layer, it increases the training cost and subsequent layer would cause its work on different input. Also, add more hidden layer would risk the process getting overfitting.

Add more nodes to hidden layer and keep other parameters the same:

Number of Hidden Layer: 1

Number of Node in hidden layer: 500

Dropout: no dropout

Batch size: 128

Epoch: 20

```
Test loss: 1.053371434671837
Test accuracy: 0.6491228342056274
```

Base on the output, add more nodes in hidden layer doesn't increase the accuracy as well. This is because the number of noes control the capability of learning any mapping function, but the learning algorithm did not realize this capability [2].

Add more epochs and keep other parameters the same:

Number of Hidden Layer: 1

Number of Node in hidden layer: 100

Dropout: no dropout

Batch size: 128

Epoch: 100

```
Test loss: 0.5611657740776999
Test accuracy: 0.8508771657943726
```

```
Epoch 1/100
 - 0s - loss: 2.6939 - accuracy: 0.2127 - val_loss: 2.3979 - val_accuracy: 0.1978
Epoch 2/100
 - 0s - loss: 2.0417 - accuracy: 0.2431 - val_loss: 1.7447 - val_accuracy: 0.3077
Epoch 3/100
 - 0s - loss: 1.7490 - accuracy: 0.2486 - val_loss: 1.7268 - val_accuracy: 0.3187
Epoch 4/100
 - 0s - loss: 1.6636 - accuracy: 0.3343 - val_loss: 1.7173 - val_accuracy: 0.2967
Epoch 5/100
 - 0s - loss: 1.6139 - accuracy: 0.3591 - val_loss: 1.6239 - val_accuracy: 0.3297
Epoch 6/100
 - 0s - loss: 1.5548 - accuracy: 0.3923 - val_loss: 1.5998 - val_accuracy: 0.4176
Epoch 7/100
 - 0s - loss: 1.5129 - accuracy: 0.4227 - val_loss: 1.5655 - val_accuracy: 0.4615
Epoch 8/100
 - 0s - loss: 1.4765 - accuracy: 0.4917 - val_loss: 1.5435 - val_accuracy: 0.3956
Epoch 9/100
 - 0s - loss: 1.4558 - accuracy: 0.4558 - val_loss: 1.5047 - val_accuracy: 0.4505
Epoch 10/100
 - 0s - loss: 1.3931 - accuracy: 0.4724 - val_loss: 1.5021 - val_accuracy: 0.4066
Epoch 11/100
 - 0s - loss: 1.3640 - accuracy: 0.5138 - val_loss: 1.4435 - val_accuracy: 0.4505
Epoch 12/100
 - 0s - loss: 1.3184 - accuracy: 0.5414 - val_loss: 1.4195 - val_accuracy: 0.4835
Epoch 13/100
 - 0s - loss: 1.2843 - accuracy: 0.5470 - val_loss: 1.4057 - val_accuracy: 0.4505
Epoch 14/100
 - 0s - loss: 1.2550 - accuracy: 0.5580 - val_loss: 1.3791 - val_accuracy: 0.4725
Epoch 15/100
 - 0s - loss: 1.2200 - accuracy: 0.5718 - val_loss: 1.3548 - val_accuracy: 0.4725
Epoch 16/100
 - 0s - loss: 1.1957 - accuracy: 0.6188 - val_loss: 1.3382 - val_accuracy: 0.4945
Epoch 17/100
 - 0s - loss: 1.1615 - accuracy: 0.6409 - val_loss: 1.3143 - val_accuracy: 0.4615
Epoch 18/100
 - 0s - loss: 1.1390 - accuracy: 0.6381 - val_loss: 1.2964 - val_accuracy: 0.5165
Epoch 19/100
 - 0s - loss: 1.1168 - accuracy: 0.6547 - val_loss: 1.2818 - val_accuracy: 0.5165
Epoch 20/100
```

Base on the output, add more epochs would significantly increase accuracy, because the accuracy is increasing overall (see epoch from 1 to 20). Also, adding more epochs would decrease the variance, so that increase the accuracy to a good balance.

Therefore, by adding more epochs, it could increase the accuracy to a consistent value.

Decrease the batch size and keep other parameters the same:

Number of Hidden Layer: 1

Number of Node in hidden layer: 100

Dropout: no dropout

Batch size: 20

Epoch: 20

```
Test loss: 0.7965596621496636
Test accuracy: 0.6842105388641357
```

Base on the output, decrease the batch size also increase the accuracy. Because with small number of batch size, it goes through the system more quickly with less variability, which fosters faster learning [1].

Therefore, the best model is:

Number of Hidden Layer: 1

Number of Node in hidden layer: 100

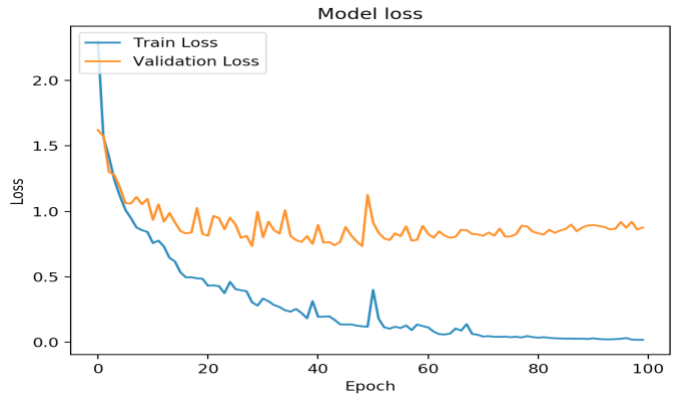
Dropout: no dropout

Batch size: 20

Epoch: 100

```
Test loss: 0.6045644973453722
Test accuracy: 0.8070175647735596
```

With Lose graph:



References


1. https://www.scaledagileframework.com/visualize-and-limit-wip-reduce-batch-sizes-and-manage-queue-lengths/
2. https://machinelearningmastery.com/how-to-control-neural-network-model-capacity-with-nodes-and-layers/