Deep Learning

Assignment 4

Multiclass Classification

Abstract

To perform multi class classification on CIFAR10 dataset using convolution neural networks and to hyper tune the model to get the best accuracy. Finally inception and residual models are added to check if they can enhance the accuracy more.

Problem Statement

The CIFAR10 dataset has 10 classes of images. Each class has 6000 images of which 5000 are for training and 1000 are for testing. The images are of dimensions 32x32x3 which are to be classified by a multi class classifier. These images are to be preprocesses, normalized and the label had to be created. Finally the convent model is built, hyper tuned and the best model is proceeded with inception and residual blocks to check their performance.

Proposed Solution

Data preprocessing

The dataset is chosen and preprocessed. This process includes converting the images to numerical forms, splitting the dataset into test, train and validation parts.

Model layers

The convent layers are used in the model as specified in the question. The layers include:

Conv2D layer

A conv2D layer is a convolution layer that includes the number of filters and the dimensions of the filter and an activation function. In each convolution layer all the filters with the given dimensions are convoluted with the images to extract the features of the image. By this means more of features are extracted with less number of parameters.

MaxPool2D

The aim of this layer is to reduce the dimensionality of the input and extract features at the same time. For a given dimensions, it chooses the image and extracts features, combines them together. By this way it reduces the coefficients of the models.

Building the model

A convolution network is built with three layers of convolution and three layers of pooling. The number of filters in each layer and the dimensions are chosen randomly to give the best

accuracy. Finally a fully connected model with one dense layer and an output layer with a softmax activation function is used.

Activations used in the model layer

Relu activation

Relu activation function is the most commonly used activation function. This is because it converges all positive values to one and all negative values to zero. Thus it is found more optimizing than the other activations.

Softmax activation

Softmax activation is used on the final dense layer to give maximum accuracy. Softmax activation is the combination of both sigmoid and ReLU. It produces a large value for larger inputs and a smaller value for smaller inputs.

Optimizer used in the model

Rmsprop – It is a gradient based optimization technique to optimize the neural network model. It decreased the gradient that increases to stop from exploding and increases the gradient that decreases to prevent from vanishing.

Loss used in the model

Categorical cross entropy – This loss is similar to a cross entropy or a softmax function. It predicts the difference between the actual and predicted labels across all the instances.

Regularization

Regularization includes certain methods followed to prevent the data from overfitting and under fitting once the correct loss functions and optimizers are determined.

The techniques used are:

Dropout – After each iteration the weight of the input or the parameter is decayed or decreased to prevent overfitting.

Batch Normalization – In this case the output of each layer of the neural network is normalized in batches to give a scalable input to the following layers.

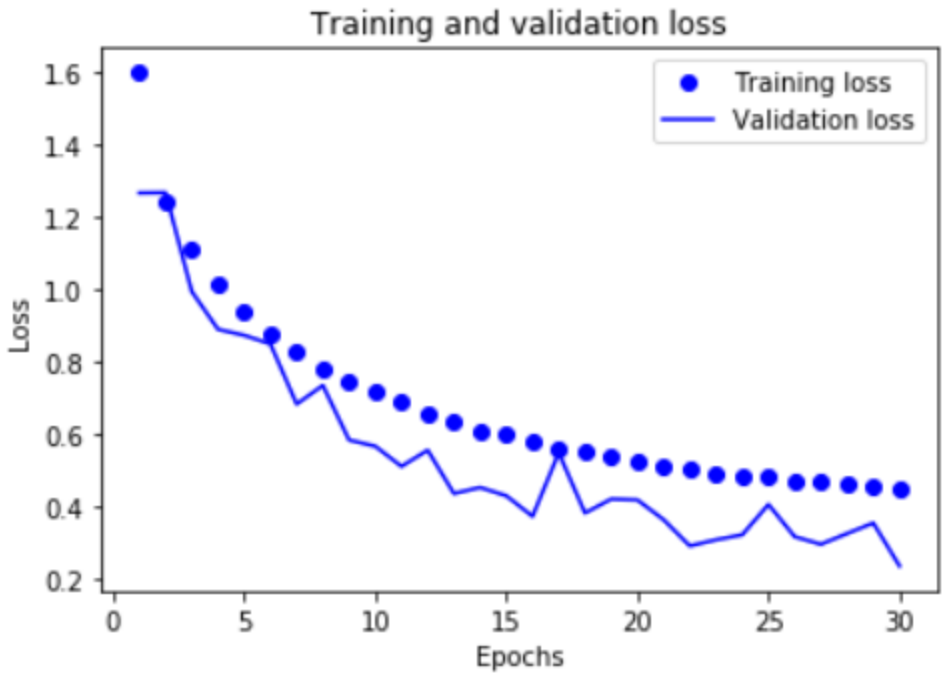
Evaluation metrics

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 15, 32)        0
```

```
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               295040
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 315,722
Trainable params: 315,722
Non-trainable params: 0
_________________________________________________________________
```

Fitting the model

```
Epoch 26/30
50000/50000 [==============================] - 61s 1ms/step - loss: 0.4716
- accuracy: 0.8371 - val_loss: 0.2810 - val_accuracy: 0.9201
Epoch 27/30
50000/50000 [==============================] - 65s 1ms/step - loss: 0.4617
- accuracy: 0.8406 - val_loss: 0.2383 - val_accuracy: 0.9239
Epoch 28/30
50000/50000 [==============================] - 63s 1ms/step - loss: 0.4595
- accuracy: 0.8421 - val_loss: 0.2439 - val_accuracy: 0.9274
Epoch 29/30
50000/50000 [==============================] - 61s 1ms/step - loss: 0.4534
- accuracy: 0.8457 - val_loss: 0.2808 - val_accuracy: 0.9173
Epoch 30/30
50000/50000 [==============================] - 61s 1ms/step - loss: 0.4499
- accuracy: 0.8450 - val_loss: 0.3685 - val_accuracy: 0.8776
```



Accuracy of test data

```
[0.9213737533569336, 0.7294999957084656]
```

=72%

Fine tuning done to the model

A dropout of 0.25 was added to the model layers to get better accuracy.

Inception model

An inception model allows multiple convolution models to be implemented at the same time and the model uses the best model. Thus this model enables the process of hyper tuning as multiple convolutions are used within the same model and thus the model choses the best layer.

Naïve version

The naïve inception model is used to determine it performance. It consists of a 3x3 convolution layer, 5x5 convolution layer, 1x1 convolution layer and a max pool 3x3 layer. The output of all these layers are concatenated to give the final output. This layer is added to the model in between the convolution and the fully connected layers. The same optimizer, loss and epochs were used for this model too.

Evaluation metrics

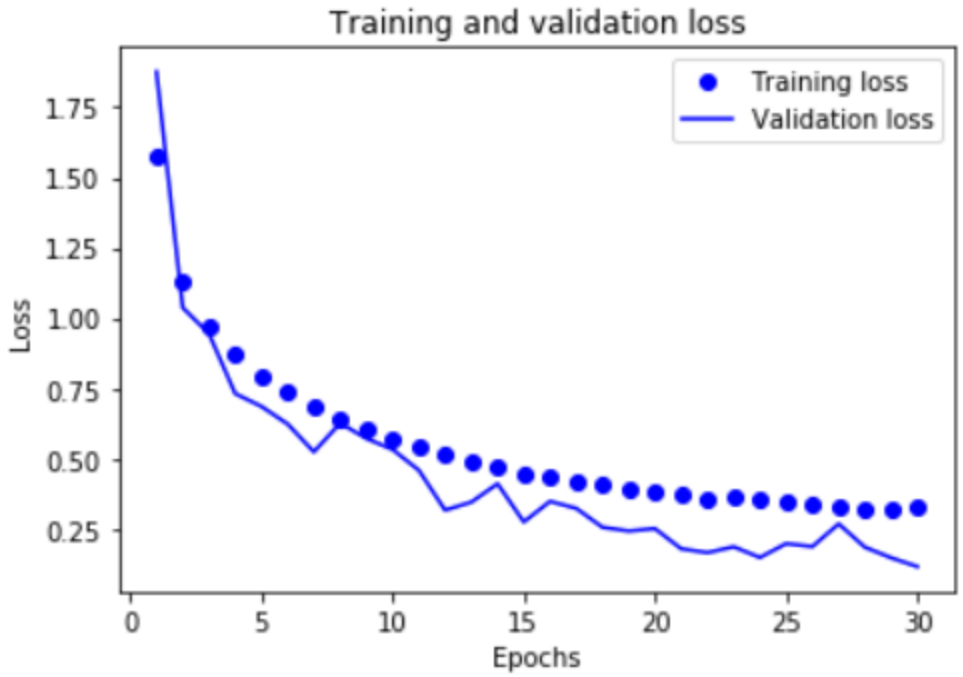
```
Model: "model_2"
__________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected
to
==================================================================================
```

```
input_2 (InputLayer)            (None, 32, 32, 3)    0
_________________________
conv2d_6 (Conv2D)               (None, 30, 30, 32)   896         input_2[0
][0]
_________________________
max_pooling2d_4 (MaxPooling2D)  (None, 15, 15, 32)   0           conv2d_6[
0][0]
_________________________
dropout_3 (Dropout)             (None, 15, 15, 32)   0           max_pooli
ng2d_4[0][0]
_________________________
conv2d_7 (Conv2D)               (None, 13, 13, 64)   18496       dropout_3
[0][0]
_________________________
max_pooling2d_5 (MaxPooling2D)  (None, 6, 6, 64)     0           conv2d_7[
0][0]
_________________________
dropout_4 (Dropout)             (None, 6, 6, 64)     0           max_pooli
ng2d_5[0][0]
_____________________
conv2d_8 (Conv2D)               (None, 6, 6, 64)     4160        dropout_4
[0][0]
_____________________
conv2d_9 (Conv2D)               (None, 6, 6, 128)    73856       dropout_4
[0][0]
_____________________
conv2d_10 (Conv2D)              (None, 6, 6, 32)     51232       dropout_4
[0][0]
_____________________
max_pooling2d_6 (MaxPooling2D)  (None, 6, 6, 64)     0           dropout_4
[0][0]
_________________________
concatenate_2 (Concatenate)     (None, 6, 6, 288)    0           conv2d_8[
0][0]
                                                                 conv2d_9[
0][0]
                                                                 conv2d_10
[0][0]
                                                                 max_pooli
ng2d_6[0][0]
_________________________
```

```
flatten_2 (Flatten)             (None, 10368)        0           concatena
te_2[0][0]
__________________________________________________________________________
_____________________
dense_3 (Dense)                 (None, 128)          1327232     flatten_2
[0][0]
__________________________________________________________________________
_____________________
dense_4 (Dense)                 (None, 10)           1290        dense_3[0
][0]
==========================================================================
=======================
Total params: 1,477,162
Trainable params: 1,477,162
Non-trainable params: 0
__________________________________________________________________________
```

Fitting the model

```
Epoch 26/30
50000/50000 [==============================] - 102s 2ms/step - loss: 0.334
5 - accuracy: 0.8880 - val_loss: 0.1868 - val_accuracy: 0.9408
Epoch 27/30
50000/50000 [==============================] - 103s 2ms/step - loss: 0.332
6 - accuracy: 0.8887 - val_loss: 0.2677 - val_accuracy: 0.9168
Epoch 28/30
50000/50000 [==============================] - 103s 2ms/step - loss: 0.322
2 - accuracy: 0.8919 - val_loss: 0.1846 - val_accuracy: 0.9394
Epoch 29/30
50000/50000 [==============================] - 103s 2ms/step - loss: 0.321
7 - accuracy: 0.8915 - val_loss: 0.1469 - val_accuracy: 0.9536
Epoch 30/30
50000/50000 [==============================] - 103s 2ms/step - loss: 0.328
2 - accuracy: 0.8916 - val_loss: 0.1155 - val_accuracy: 0.9652
```



Accuracy on the testing model

```
[1.0767808875083924, 0.7426000237464905]
```

=74%

Residual model

A residual model is similar to an inception model whereas the layers are added instead of concatenating. This model can be defined with any number of convolution or pooling and finally can be added. The residual model taken has three convolution layers whose results are finally added and then given to the next layer of the model. The same optimizer, loss and epochs are considered for the purpose of comparisons.

Evaluation

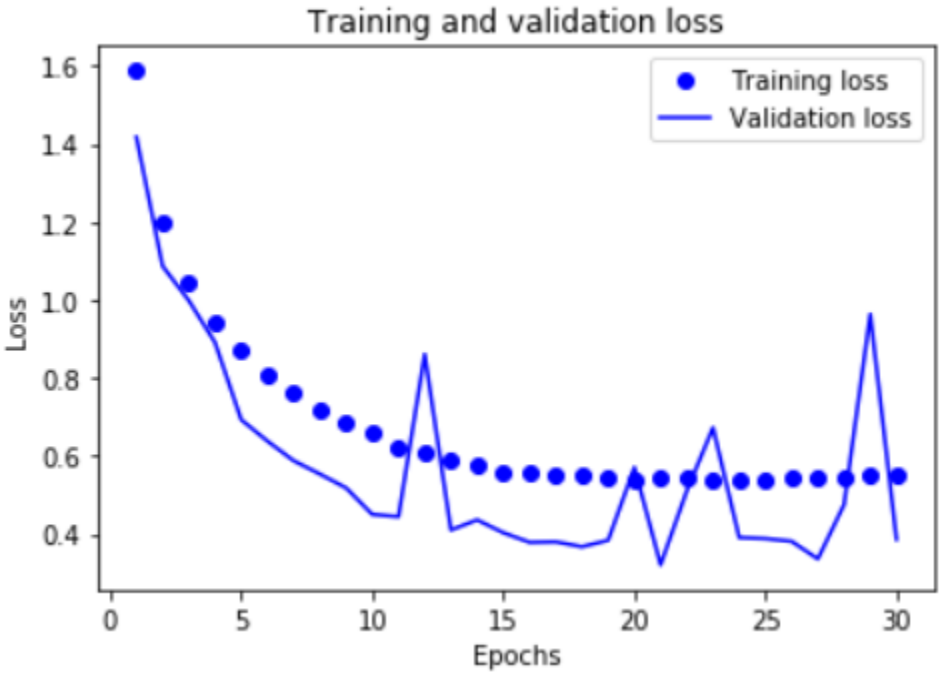
```
Model: "model_3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            (None, 32, 32, 3)    0
__________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 30, 30, 32)   896         input_4[0][0]
__________________________________________________________________________________________________
```

```
max_pooling2d_9 (MaxPooling2D)  (None, 15, 15, 32)   0           conv2d_16
[0][0]
_________________________________________________________________
_____________________________
dropout_7 (Dropout)             (None, 15, 15, 32)   0           max_pooli
ng2d_9[0][0]
_________________________________________________________________
_____________________________
conv2d_17 (Conv2D)              (None, 13, 13, 64)   18496       dropout_7
[0][0]
_________________________________________________________________
_____________________________
max_pooling2d_10 (MaxPooling2D) (None, 6, 6, 64)     0           conv2d_17
[0][0]
_________________________________________________________________
_____________________________
dropout_8 (Dropout)             (None, 6, 6, 64)     0           max_pooli
ng2d_10[0][0]
_________________________________________________________________
_____________________________
conv2d_20 (Conv2D)              (None, 6, 6, 64)     36928       dropout_8
[0][0]
_________________________________________________________________
_____________________________
add_2 (Add)                     (None, 6, 6, 64)     0           conv2d_20
[0][0]
                                                                 dropout_8
[0][0]
_________________________________________________________________
flatten_3 (Flatten)             (None, 2304)         0           add_2[0][
0]
_____________________________________________________________________
dense_5 (Dense)                 (None, 128)          295040      flatten_3
[0][0]
_____________________________________________________________________
dense_6 (Dense)                 (None, 10)           1290        dense_5[0
][0]
=================================================================
Total params: 352,650
Trainable params: 352,650
Non-trainable params: 0
```

Fitting the model

```
Epoch 26/30
50000/50000 [==============================] - 35s 705us/step - loss: 0.54
33 - accuracy: 0.8167 - val_loss: 0.3815 - val_accuracy: 0.8627
Epoch 27/30
50000/50000 [==============================] - 35s 692us/step - loss: 0.54
46 - accuracy: 0.8163 - val_loss: 0.3364 - val_accuracy: 0.8937
Epoch 28/30
50000/50000 [==============================] - 34s 675us/step - loss: 0.54
70 - accuracy: 0.8161 - val_loss: 0.4757 - val_accuracy: 0.8417
Epoch 29/30
```

```
50000/50000 [==============================] - 35s 699us/step - loss: 0.55
01 - accuracy: 0.8162 - val_loss: 0.9634 - val_accuracy: 0.6850
Epoch 30/30
50000/50000 [==============================] - 36s 713us/step - loss: 0.55
27 - accuracy: 0.8152 - val_loss: 0.3864 - val_accuracy: 0.8773
```



Accuracy on the test model

```
[0.8636185321807861, 0.7178000211715698]
```

=72%

Conclusion

From the above models and comparisons it is obvious that the inception and residual models produce more accuracy and less over fitting than the normal convolution models.

Implementation details

Implemented using anaconda and jupyter.

The issues included

1. Large datasets took a longer time.
2. Time consumption due to evaluation against all models.
3. The code was ran on a CPU that consumed more time.
4. Not every model was saved because the models were misinterpreted by the jupyter kernel. Thus only the best model code is provided.

References:

1. Code given by the professor