Name: Rahman Farzad

Student ID: 22-92350-1

Introduction

CIFAR-10 is an established computer-vision dataset used for object recognition. It is a subset of the 80 million tiny images dataset and consists of 60,000 32x32 color images containing one of 10 object classes, with 6000 images per class. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

Here are the classes in the dataset, as well as 10 random images from each:
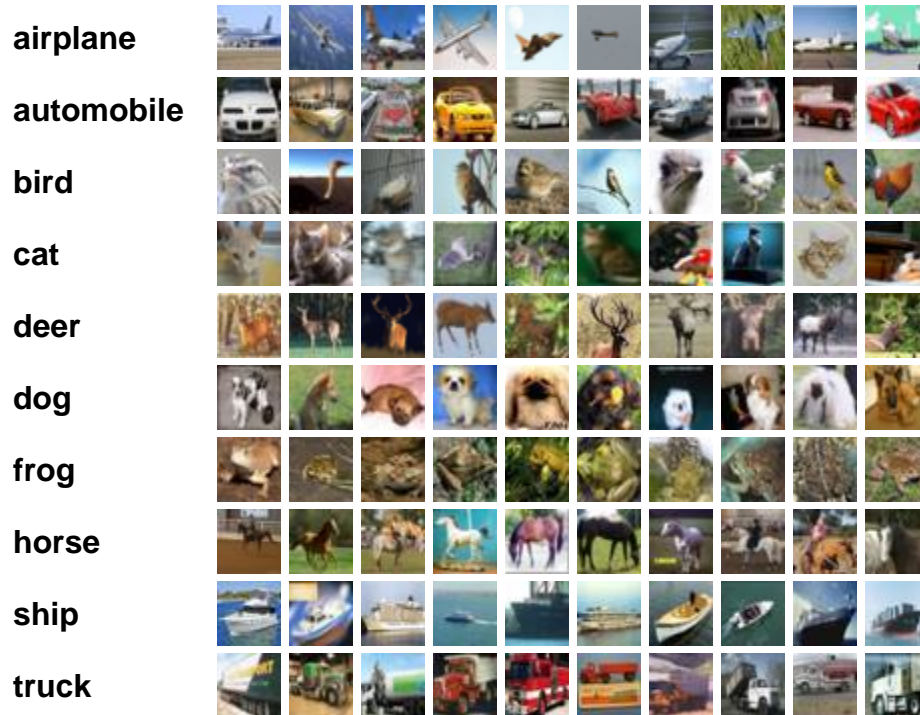
Here I have designed a network that combines supervised and unsupervised architectures in one model to achieve a classification on CIFAR-10 datasets. So, for this I build a encoder model and this model compressed the image data and after decoder model decompressed the data and again this model reconstruct the original image.

Another part of this notebook is to use the Pre-Training CNNs Using Convolutional Autoencoders to classify the image.

For completing this project/ assignment I have followed some module those are:

- Tensorflow
- Keras
- Scikit-learn
- Pandas
- Numpy

- cifar10
- to_categorical

Splitting Datasets into four parts

1. train_images
2. train_labels
3. test_images
4. test_labels

It is clear that the images of the datasets are indeed very small compared to modern photographs; it can be challenging to see what exactly is represented in some of the images given the extremely low Untitled 12/3/22 11:42 PM resolution. This low resolution is likely the cause of the limited performance that top-of-the-line algorithms are able to achieve on the dataset. The max pixel value is 255 for each channel. Normalize the images to a number from 0 to 1. Image has 3 channels (R,G,B) and each value in the channel can range from 0 to 255. Hence to normalize in 0-->1 range, we need to divide it by 255

I have used 3 different optimizers into my task for compare the accuracy and loss values

1. Adam
2. Rmsprop
3. SGD

And 3 loss functions:

1. *sparse_categorical_crossentropy*
2. categorical_crossentropy
3. binary_crossentropy

Cifer10 Dataset Using ADAM optimizer with sparse_categorical_crossentropy loss Function

Summary of the model:

```
Model: "sequential"
_____
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        2432
_____
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_____
conv2d_1 (Conv2D)            (None, 12, 12, 64)        18496
_____
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_____
conv2d_2 (Conv2D)            (None, 4, 4, 128)         73856
_____
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 128)         0
_____
flatten (Flatten)            (None, 512)               0
_____
dense (Dense)                (None, 128)               65664
_____
dense_1 (Dense)              (None, 64)                8256
_____
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 169,354
Trainable params: 169,354
Non-trainable params: 0
```
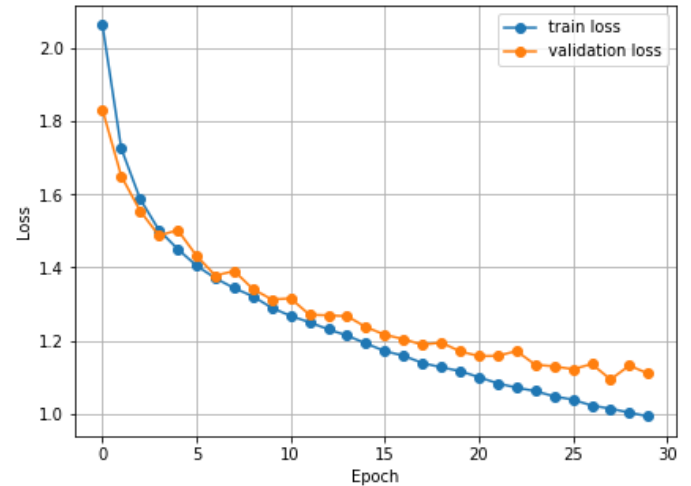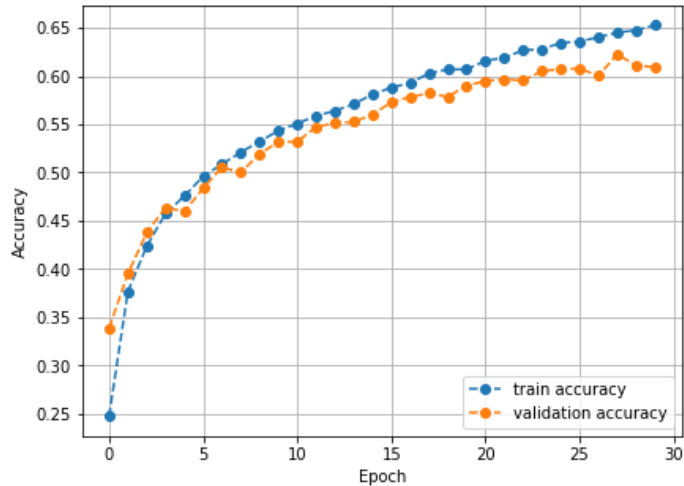
Training the model:

model**.**fit(x=train_images_norm,y=train_labels,epochs=30,batch_size=128,validation_split=0.3)

number of epochs was 30 and batch size=128

Data accuracy & loss chart:

After evaluating the model
Model Losses: 1.0952
Model accuracy: 0.6175

Cifer10 DataSet Using RMSPROP optimizer with categorical_crossentropy loss Function
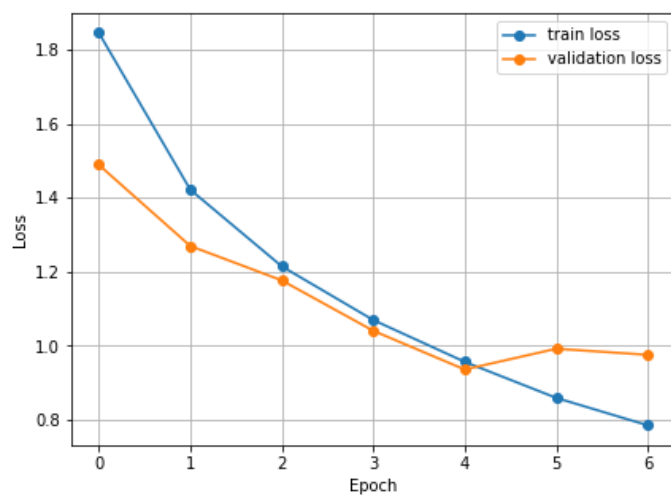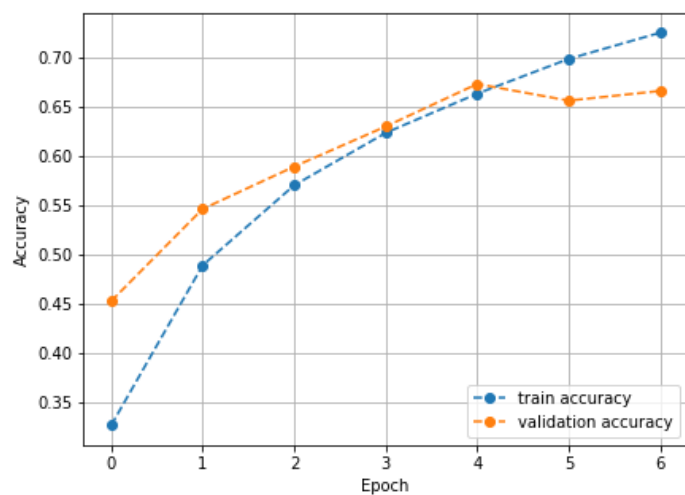
Summary of the model:

```
Model: "sequential_2"
_____
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_7 (Conv2D)            (None, 28, 28, 32)        2432

 max_pooling2d_5 (MaxPooling  (None, 14, 14, 32)        0
 2D)

 conv2d_8 (Conv2D)            (None, 12, 12, 64)        18496

 max_pooling2d_6 (MaxPooling  (None, 6, 6, 64)          0
 2D)

 conv2d_9 (Conv2D)            (None, 4, 4, 128)         73856

 max_pooling2d_7 (MaxPooling  (None, 2, 2, 128)         0
 2D)

 flatten_2 (Flatten)          (None, 512)               0

 dense_6 (Dense)              (None, 128)               65664

 dense_7 (Dense)              (None, 256)               33024

 dense_8 (Dense)              (None, 10)                2570

=================================================================
Total params: 196,042
Trainable params: 196,042
Non-trainable params: 0
_____
```

Training the model:

h=model**.**fit(train_images_norm, y_train_en, batch_size=128, epochs=20, validation_data=(test_images_norm, y_test_en),callbacks=callbacks)

number of epochs was 20 and batch size=128

Data accuracy & loss chart:

After evaluating the model

Model Losses: 0.9363400340080261
Model accuracy: 0.6733999848365784

Cifer10 DataSet Using SGD optimizer with binary_crossentropy loss Function

Summary of the model:

```
Model: "sequential_3"
_____
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 28, 28, 32)        2432

 max_pooling2d_9 (MaxPooling  (None, 14, 14, 32)        0
 2D)

 conv2d_10 (Conv2D)          (None, 12, 12, 64)        18496

 max_pooling2d_10 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_11 (Conv2D)          (None, 4, 4, 128)         73856

 max_pooling2d_11 (MaxPoolin  (None, 2, 2, 128)         0
 g2D)

 flatten_3 (Flatten)         (None, 512)               0

 dense_9 (Dense)             (None, 128)               65664

 dense_10 (Dense)            (None, 256)               33024

 dense_11 (Dense)            (None, 10)                2570

=================================================================
Total params: 196,042
Trainable params: 196,042
Non-trainable params: 0
_____
```
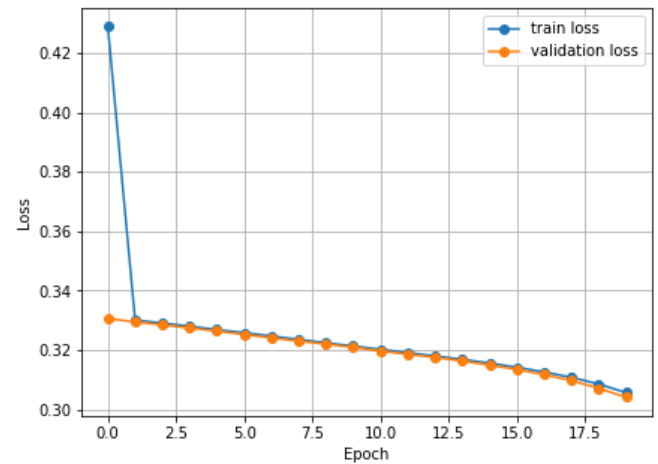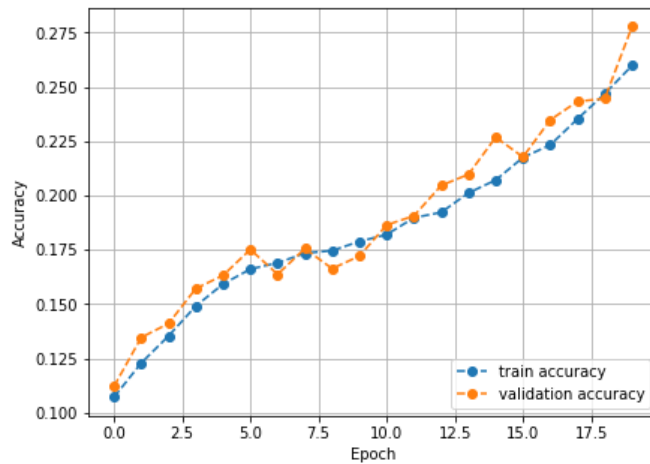
Training the model:

```
h=model.fit(train_images_norm, y_train_en, batch_size=128, epochs=20,
validation_data=(test_images_norm, y_test_en),callbacks=callbacks)
```

```
number of epochs was 20 and batch size=128
```


After evaluating the model

```
Model Losses:  0.3042064309120178
Model accuracy:  0.2777999937534332
```

Summary:

Optimizers Name	Loss	Accuracy
Adam	1.0952	0.6175
RmsProp	0.9363	0.6734
SGD	0.3042	0.2778

I have got different accuracy metrics for different optimizers using condition.