Assignment 6.2a

In [1]:
```
import keras
import tensorflow as tf
from keras.datasets import cifar10

from keras import layers
from keras import models
import matplotlib.pyplot as plt
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
```

```
Using TensorFlow backend.
```

Load Data Set

In [2]:
```
# Loading the data into memory
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

In [3]:
```
x_train.shape
```

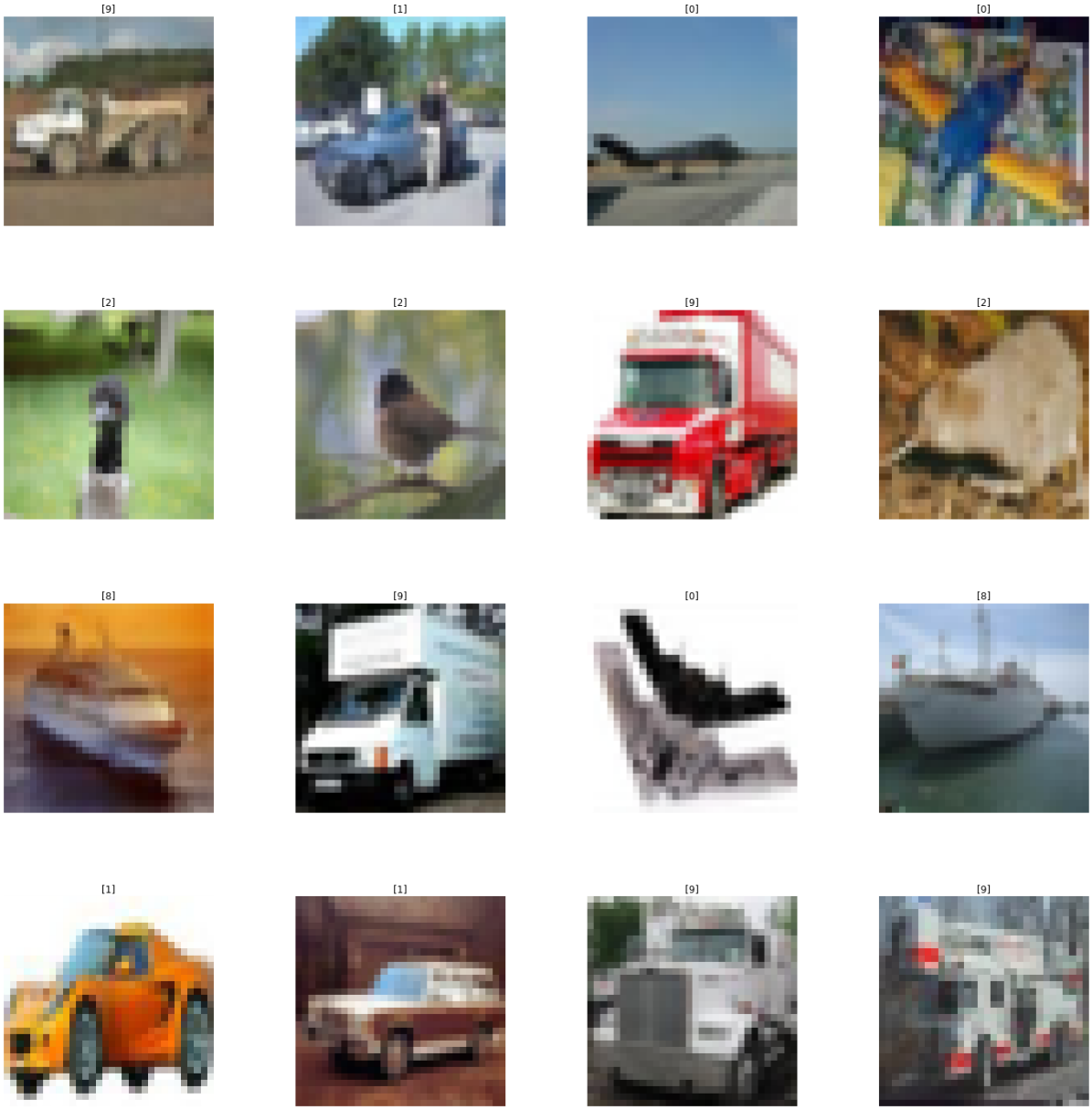
Out[3]:
```
(50000, 32, 32, 3)
```

In [4]:
```
y_train.shape
```

Out[4]:
```
(50000, 1)
```

In [5]:
```
W_grid = 4
L_grid = 4
fig, axes = plt.subplots(L_grid, W_grid, figsize = (25, 25))
axes = axes.ravel()
n_training = len(x_train)
for i in np.arange(0, L_grid * W_grid):
    index = np.random.randint(0, n_training) # pick a random number
    axes[i].imshow(x_train[index])
    axes[i].set_title(y_train[index])
    axes[i].axis('off')
plt.subplots_adjust(hspace = 0.4)
```

```
C:\Users\bibek\anaconda3\envs\dsc650\lib\site-packages\matplotlib\text.py:1150: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  if s != self._text:
```



Data Preparation

In [6]:
```
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

y_train = y_train.reshape(y_train.shape[0])
y_test = y_test.reshape(y_test.shape[0])
```

Validation Dataset

In [7]:
```
x_val_train = x_train[:10000]
x_train = x_train[10000:]

y_val_train = y_train[:10000]
y_train = y_train[10000:]
```

Instantiating a Small Covnet

In [8]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', padding = 'same',
                        input_shape=x_train.shape[1:]))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='sigmoid'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 5, 128)         73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 361,034
Trainable params: 361,034
Non-trainable params: 0
_________________________________________________________________
```

In [9]:
```
from keras import optimizers
```

Building a Neural Network

In [10]:
```
model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Training the neural network with partial_x_train and partial_y_train
history = model.fit(x_train,
                    y_train,
                    epochs = 20,
                    batch_size = 128,
                    validation_data = (x_val_train, y_val_train),
                    verbose=True)
```

```
Train on 40000 samples, validate on 10000 samples
Epoch 1/20
40000/40000 [==============================] - 21s 535us/step - loss: 2.0406
- accuracy: 0.2572 - val_loss: 1.9585 - val_accuracy: 0.2711
Epoch 2/20
40000/40000 [==============================] - 22s 555us/step - loss: 1.7731
- accuracy: 0.3532 - val_loss: 1.7868 - val_accuracy: 0.3545
Epoch 3/20
40000/40000 [==============================] - 23s 568us/step - loss: 1.6595
- accuracy: 0.3988 - val_loss: 1.6384 - val_accuracy: 0.3896
Epoch 4/20
40000/40000 [==============================] - 23s 565us/step - loss: 1.5841
- accuracy: 0.4273 - val_loss: 1.6020 - val_accuracy: 0.4266
Epoch 5/20
40000/40000 [==============================] - 23s 571us/step - loss: 1.5263
- accuracy: 0.4488 - val_loss: 1.4835 - val_accuracy: 0.4588
Epoch 6/20
40000/40000 [==============================] - 23s 567us/step - loss: 1.4794
- accuracy: 0.4675 - val_loss: 1.5183 - val_accuracy: 0.4457
Epoch 7/20
40000/40000 [==============================] - 23s 571us/step - loss: 1.4357
- accuracy: 0.4892 - val_loss: 1.4310 - val_accuracy: 0.4912
Epoch 8/20
40000/40000 [==============================] - 23s 573us/step - loss: 1.3988
- accuracy: 0.5029 - val_loss: 1.3958 - val_accuracy: 0.4993
Epoch 9/20
40000/40000 [==============================] - 23s 574us/step - loss: 1.3655
- accuracy: 0.5174 - val_loss: 1.3653 - val_accuracy: 0.5070
Epoch 10/20
40000/40000 [==============================] - 23s 576us/step - loss: 1.3389
- accuracy: 0.5298 - val_loss: 1.3489 - val_accuracy: 0.5270
Epoch 11/20
40000/40000 [==============================] - 23s 576us/step - loss: 1.3114
- accuracy: 0.5385 - val_loss: 1.2929 - val_accuracy: 0.5456
Epoch 12/20
40000/40000 [==============================] - 24s 593us/step - loss: 1.2879
- accuracy: 0.5466 - val_loss: 1.4090 - val_accuracy: 0.5040
Epoch 13/20
40000/40000 [==============================] - 24s 592us/step - loss: 1.2605
- accuracy: 0.5585 - val_loss: 1.3054 - val_accuracy: 0.5334
Epoch 14/20
40000/40000 [==============================] - 23s 579us/step - loss: 1.2408
- accuracy: 0.5646 - val_loss: 1.3223 - val_accuracy: 0.5295
Epoch 15/20
40000/40000 [==============================] - 24s 588us/step - loss: 1.2183
- accuracy: 0.5738 - val_loss: 1.2371 - val_accuracy: 0.5611
Epoch 16/20
40000/40000 [==============================] - 25s 636us/step - loss: 1.1993
- accuracy: 0.5814 - val_loss: 1.2090 - val_accuracy: 0.5710
Epoch 17/20
40000/40000 [==============================] - 27s 684us/step - loss: 1.1812
- accuracy: 0.5889 - val_loss: 1.2005 - val_accuracy: 0.5681
Epoch 18/20
40000/40000 [==============================] - 27s 677us/step - loss: 1.1643
- accuracy: 0.5945 - val_loss: 1.1959 - val_accuracy: 0.5809
Epoch 19/20
40000/40000 [==============================] - 24s 596us/step - loss: 1.1461
```

```
- accuracy: 0.6010 - val_loss: 1.2313 - val_accuracy: 0.5581
Epoch 20/20
40000/40000 [==============================] - 28s 703us/step - loss: 1.1307
- accuracy: 0.6061 - val_loss: 1.1454 - val_accuracy: 0.5972
```

In [11]:
```
results = model.evaluate(x_test, y_test)

results
```

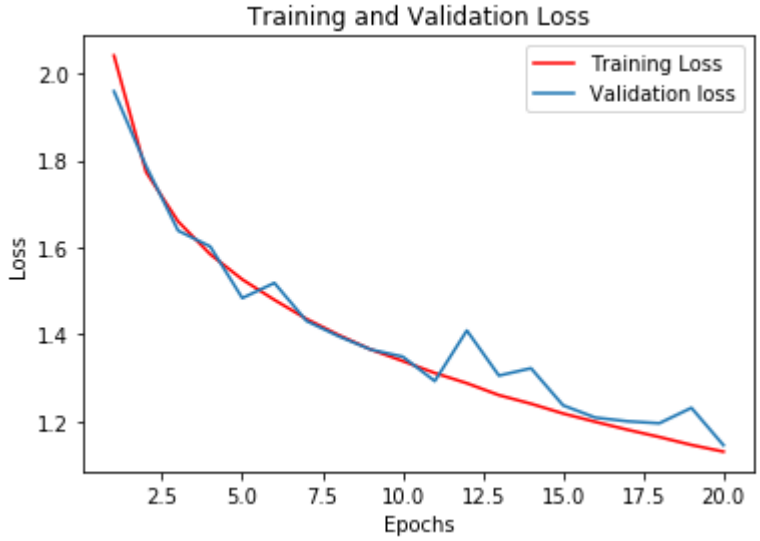
```
10000/10000 [==============================] - 2s 218us/step
```

Out[11]:
```
[1.1482285236358643, 0.5976999998092651]
```

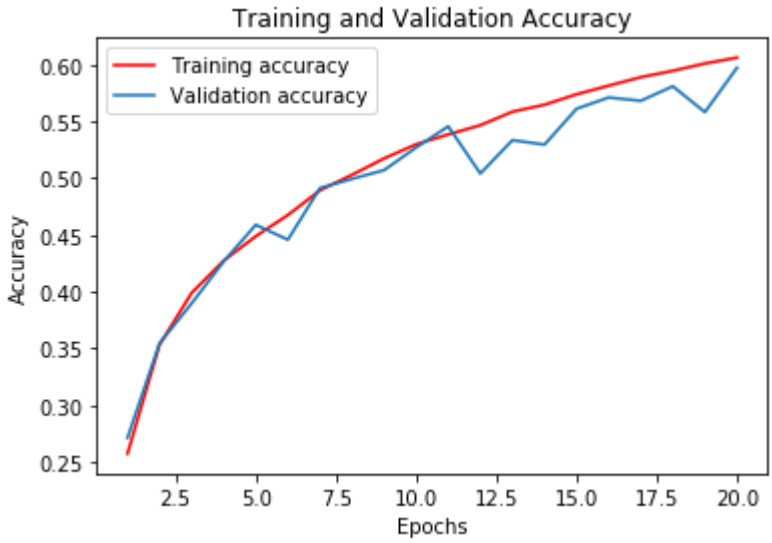
In [12]:
```
history_dict = history.history
history_dict.keys()
```

Out[12]:
```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```

In [13]:
```
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1,len(acc)+ 1)

plt.plot(epochs, loss_values,  "r-", label = 'Training Loss')
plt.plot(epochs, val_loss_values, label = 'Validation loss')
plt.title('Training and Validation Loss')
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



In [14]:
```
plt.plot(epochs, acc,  "r-", label = 'Training accuracy')
plt.plot(epochs, val_acc, label = 'Validation accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



Assignment 6.2b

In [15]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', padding = 'same',
                        input_shape=x_train.shape[1:]))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='sigmoid'))

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 5, 5, 128)         73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_4 (Dense)              (None, 10)                5130
=================================================================
Total params: 361,034
Trainable params: 361,034
Non-trainable params: 0
_________________________________________________________________
```

In [16]:
```
model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

train_datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

train_datagen.fit(x_train)

# Training the neural network with partial_x_train and partial_y_train
history = model.fit(train_datagen.flow(x_train, y_train,
                                       batch_size=128),epochs=40,
                    validation_data=(x_val_train, y_val_train),
                    workers=4,verbose=False)
```

In [17]:
```
results = model.evaluate(x_test, y_test)

results
```

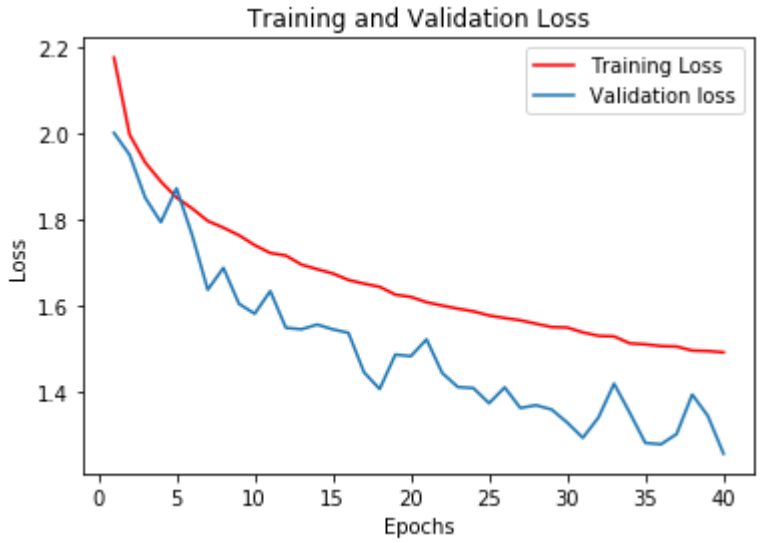
```
10000/10000 [==============================] - 3s 255us/step
```

Out[17]:
```
[1.2589075031280517, 0.5558000206947327]
```

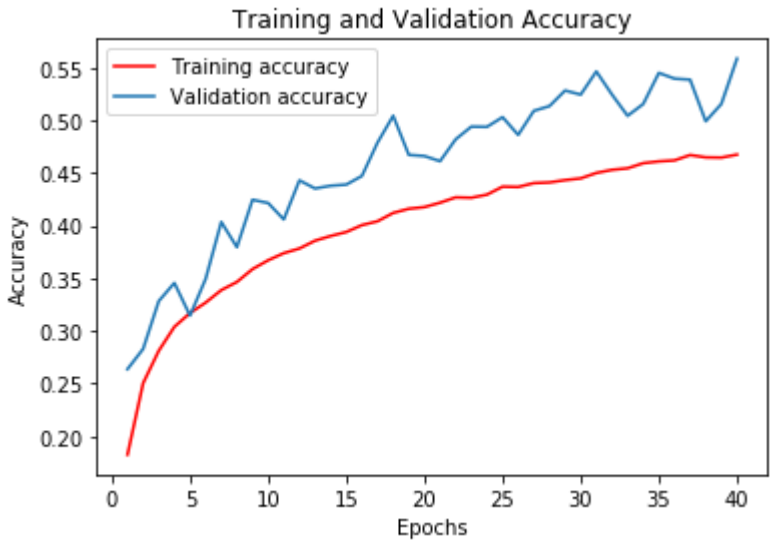
In [18]:
```
history_dict = history.history
history_dict.keys()
```

Out[18]:
```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```

In [19]:
```
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1,len(acc)+ 1)

plt.plot(epochs, loss_values,  "r-", label = 'Training Loss')
plt.plot(epochs, val_loss_values, label = 'Validation loss')
plt.title('Training and Validation Loss')
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



In [20]:
```
plt.plot(epochs, acc,  "r-", label = 'Training accuracy')
plt.plot(epochs, val_acc, label = 'Validation accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



In [ ]:
```
```