Tutorial 4

CIFAR Type Prediction Challenge

Group members :-

1) Ganapathi Bharath Manapragada
2) Kevin Cyriac Edampurath
3) Gandlur Phani Shankar Bharadwaj

Tasks

- Train a neural network to accomplish the following tasks:
- To recognise CIFAR-10 type images and categorise them into three different classes:
  - Cats
  - Dogs
  - Frogs
- Try to achieve at least 65% accuracy

Background – CIFAR 10

- The original dataset has the following properties:
  - 60000 images in total
  - Each image has size 32 x 32 pixels and 3 colour channels
  - Images can be categorized into 10 classes
  - 6000 images per class
  - 50000 training images and 10000 test images

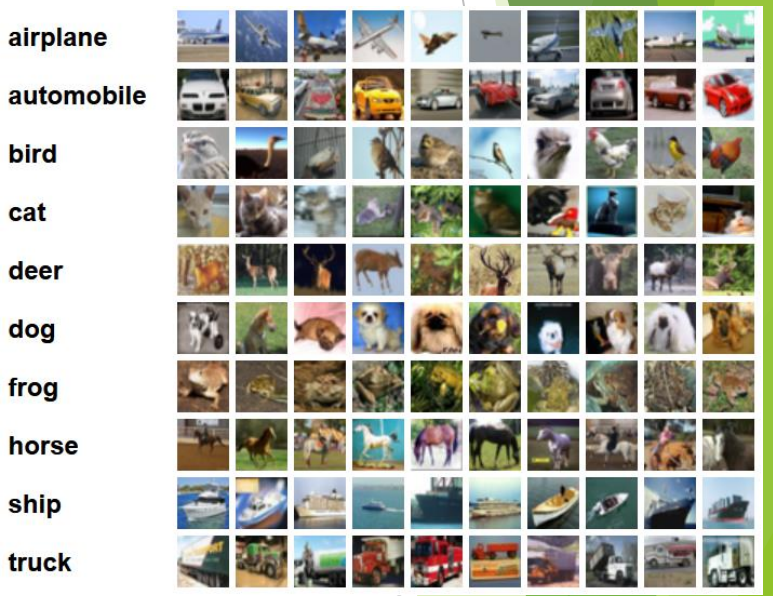

Image 1: CIFAR dataset [1]

What we were given

- Training set of 3000 images
- Categorized into three classes:
  - Cats - 0
  - Dogs - 1
  - Fogs - 2
- Test set of 300 images without labels

Background – Using Neural Networks for Image Recognition

- Image recognition is a complex task
- Feed forward networks :
  - Easy to implement
  - Computationally less demanding
  - Sufficient for handwritten character identification
  - Insufficient for CIFAR type images
- Typically used neural network – convolutional networks

Feed forward implementation

- A feed forward neural network to fulfil the task was initially implemented
- Python code used to implement it can be found in the next slide
- Problems:
  - Low training accuracy for low number of epochs
  - High number of epochs can achieve high training accuracy but overfits model
  - Hence validation accuracy is always low
- Easy to run

Feed Forward Code

```
import tensorflow as tf


import numpy as np

with np.load('prediction-challenge-02-data.npz') as fh:
    data_x = fh['x_train']
    data_y = fh['y_train']
    test_x = fh['x_test']


data_x_1 = data_x[0:2500]
data_y_1 = data_y[0:2500]

test_x_1 = data_x[2500:]
test_y_1 = data_y[2500:]

train_DS = tf.data.Dataset.from_tensor_slices((data_x_1,data_y_1))

batch_s = 2500

train_DS = train_DS.batch(batch_s)


network = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(32, 32, 3)),
  tf.keras.layers.Dense(300,activation='relu'),
  tf.keras.layers.Dense(64,activation='relu'),
  tf.keras.layers.Dense(3, activation='softmax')
])
network.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

network.fit(train_DS, validation_data=(test_x_1, test_y_1), epochs=6)
```

Feed Forward Code – Results

- After 6 Epochs:
  - Training accuracy – 33.4 %
  - Validation accuracy – 36.4%
- After 15 Epochs
  - Training accuracy – 39.64 %
  - Validation accuracy – 42%
- Can go above 65% training accuracy after a few hundred epochs. But the model is severely overfitted.

Why feed forward doesn't work

- Feed Forward networks ignore spatial configuration in the image

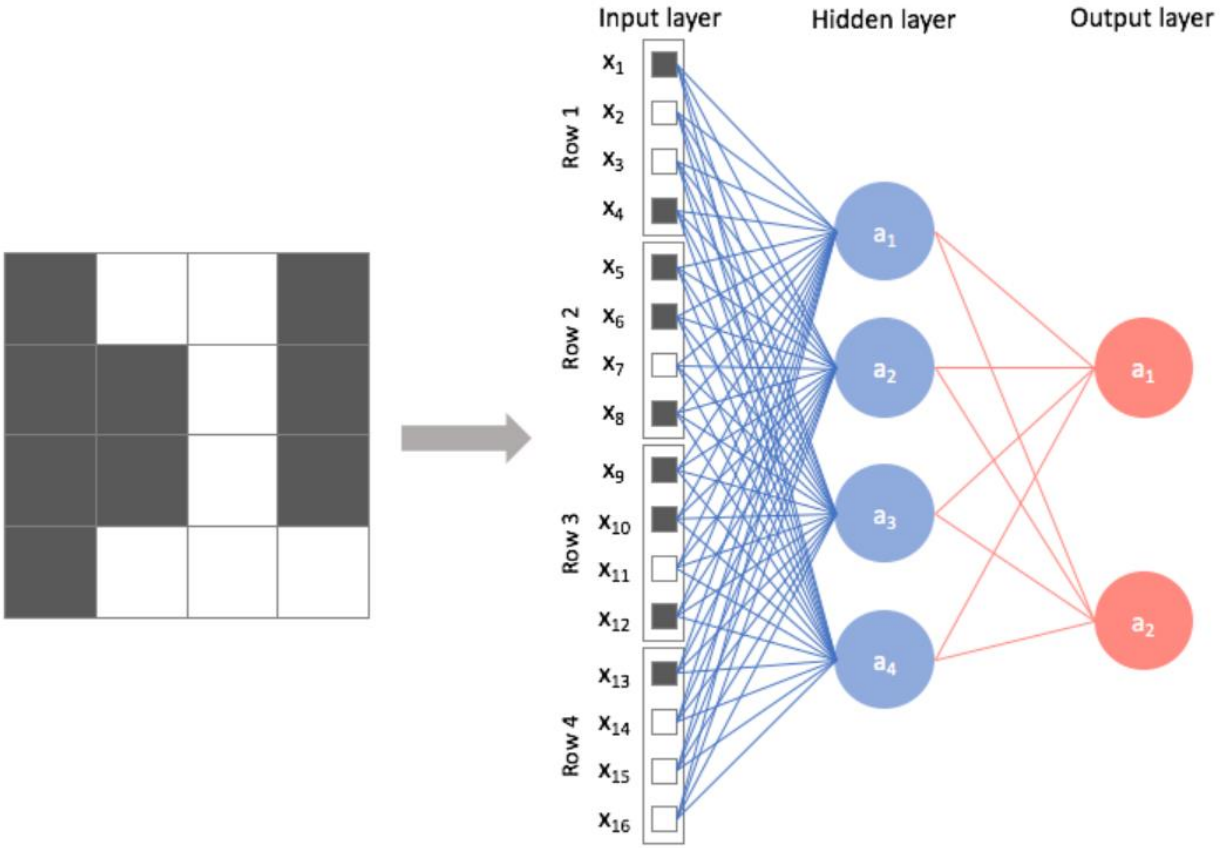

Image 2. [2]

Convolutional Neural Networks

- Convolutional Neural Networks consider the local regions within the image

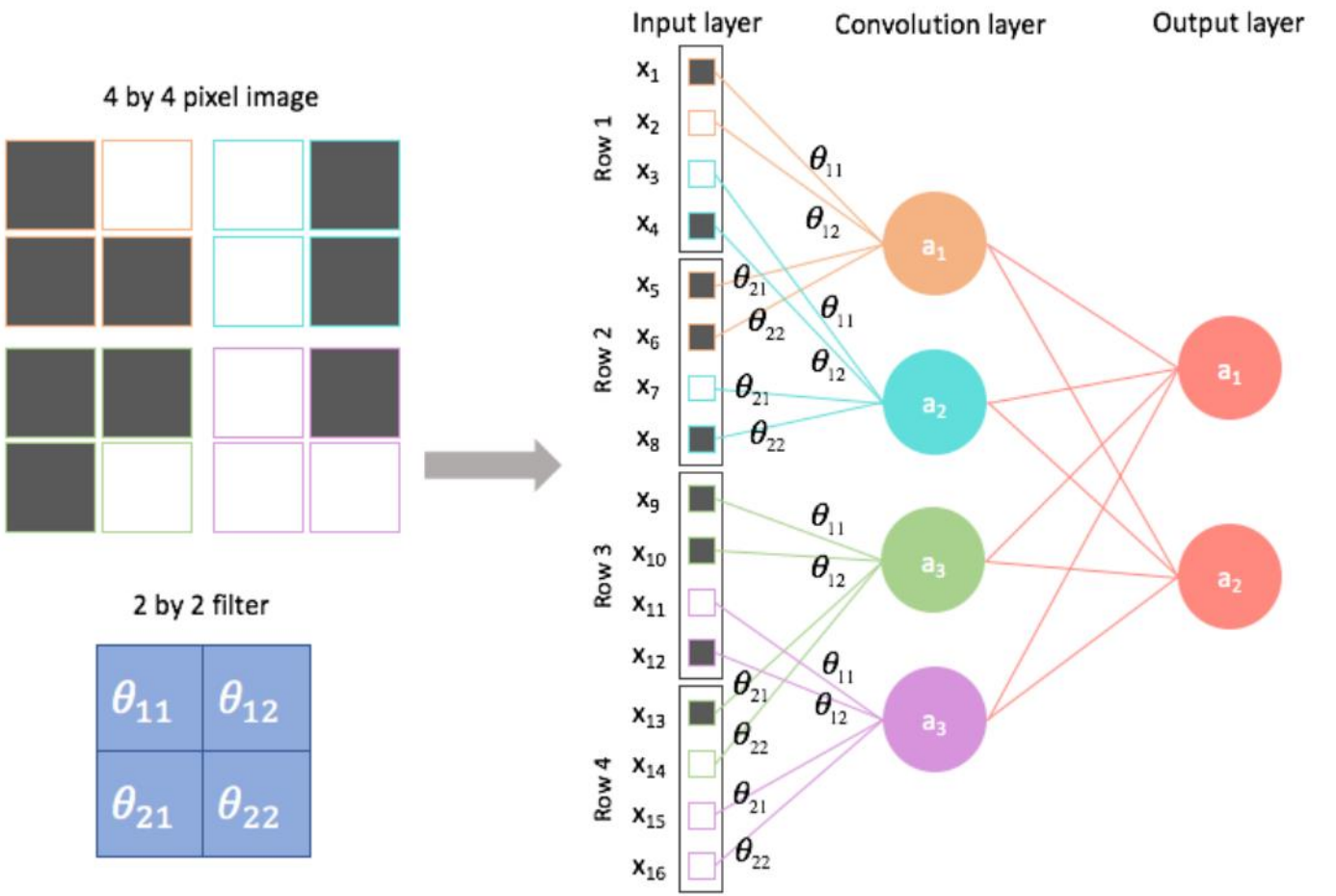

Image 3. [3]

Convolutional Neural Networks

Image 4. [4]

Pooling

- Reduces the size of the feature.
- Two common types: Max Pooling and Avg Pooling

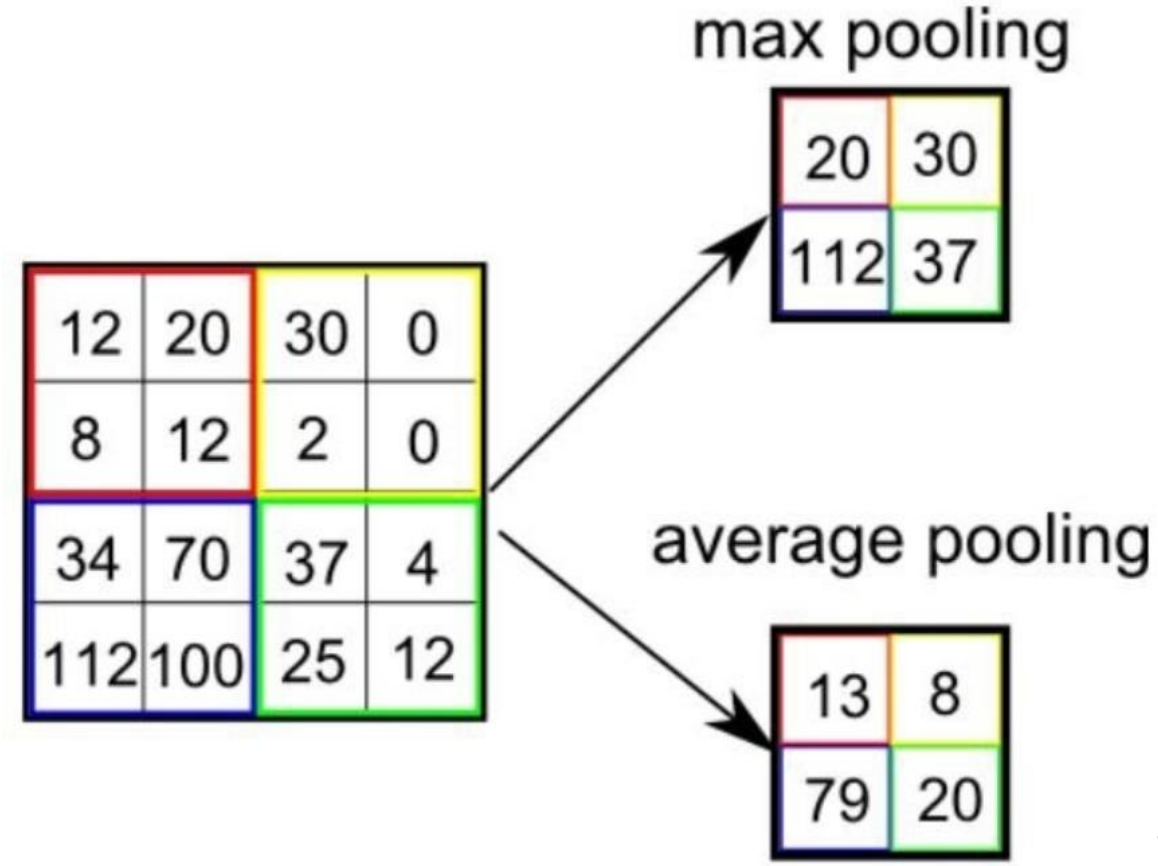

Image 5. [5]

CNN Code - 1

```
 8
 9 from keras.models import Sequential
10 from keras.layers import Conv2D
11 from keras.layers import MaxPooling2D
12 from keras.layers import Dense
13 from keras.layers import Flatten
14 from keras.layers import Dropout
15
16 import numpy as np
17
18 with np.load('prediction-challenge-02-data.npz') as fh:
19     x_train = fh['x_train']
20     y_train = fh['y_train']
21     x_test = fh['x_test']
22
23
24 # TRAINING DATA: INPUT (x) AND OUTPUT (y)
25 # 1. INDEX: IMAGE SERIAL NUMBER (6000)
26 # 2/3. INDEX: PIXEL VALUE (32 x 32)
27 # 4. INDEX: COLOR CHANNELS (3)
28 #print(x_train.shape, x_train.dtype)
29
30 x_train_1 = x_train[0:2400]
31 y_train_1 = y_train[0:2400]
32 x_test_1 = x_train[2400:]
33 y_test_1 = y_train[2400:]
34
35
36 x_train_1 = x_train_1.astype('float32')
37 x_test_1 = x_test_1.astype('float32')
38
39 x_train_1 = x_train_1/255
40 x_test_1 = x_test_1/255
41
```

CNN Code - 2

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(32, 32, 3)))
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dropout(0.2))
model.add(Dense(3, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])


history = model.fit(x_train_1, y_train_1, epochs=130, batch_size=2400)
accuracy = model.evaluate(x_test_1, y_test_1)
print(accuracy * 100.0)
```

CNN – Results

- After 50 Epochs:
  - Training accuracy – 57%
  - Validation accuracy – 54%
- After 130 Epochs
  - Training accuracy – 65%
  - Validation accuracy – 66%

Image Augmentation

- CNNs are not rotation invariant
- The size of the training set can be increased by rotating the images

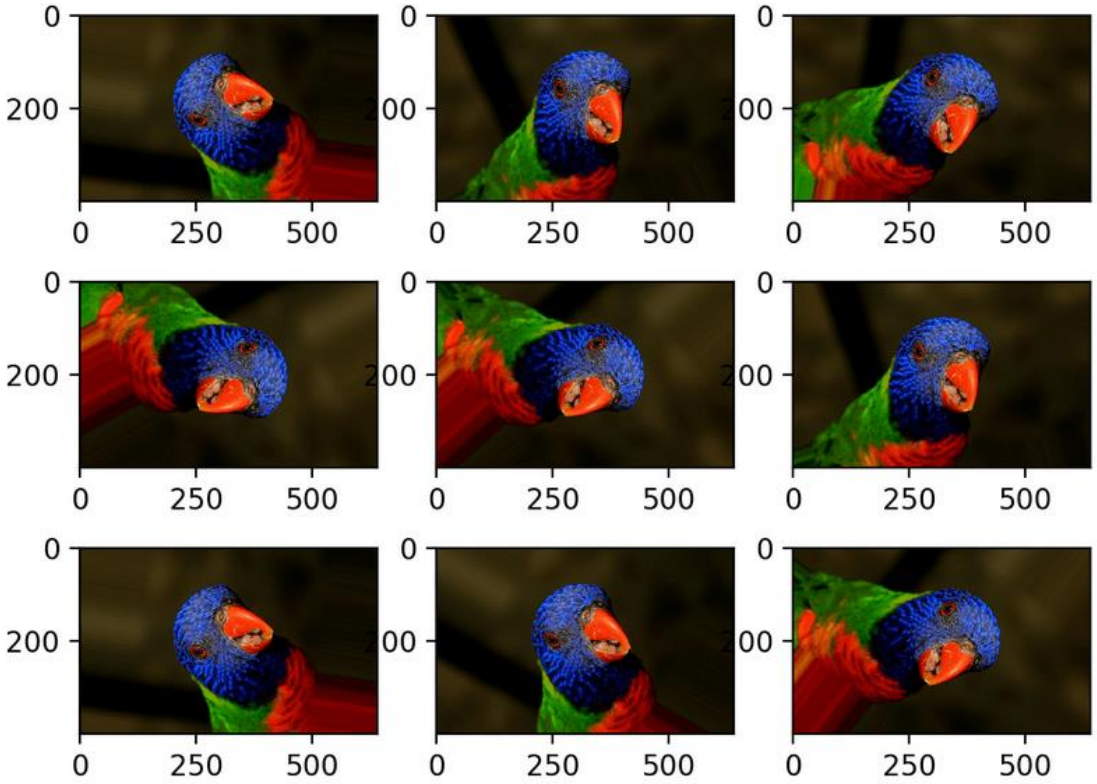

Image 6. [6]

Conclusion

- Feed Forward networks are not ideal for predicting CIFAR type images.
- Convolutional Neural Networks are much better suited for this task.
- They can be further improved by image augmentation.

References

- [1]https://www.cs.toronto.edu/~kriz/cifar.html
- [2]https://www.jeremyjordan.me/convolutional-neural-networks/
- [3]https://www.youtube.com/watch?v=zfiSAzpy9NM
- [4]https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
- [5]https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks
- [6]https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/