CODING : Influence of Network Depth and Regularization Techniques

```
# Enable GPU: "Runtime"-->"Change Runtime"-->"Hardware Accelerator"
#1.·Check·if·GPU·is·enabled
import·tensorflow·as·tf
tf.test.gpu_device_name()
```

```
'/device:GPU:0'
```

```
#2.·Import·dataset
from·keras.datasets·import·cifar10
import·numpy·as·np
from·sklearn.model_selection·import·train_test_split

(X_train,·y_train),·(X_test,·y_test)·=·cifar10.load_data()

from·tensorflow.keras.utils·import·to_categorical

y_train·=·to_categorical(y_train)
y_test·=·to_categorical(y_test)
#X_train,·X_test,·y_train,·y_test·=·train_test_split(x,·y,·train_size=0.8)

X_train,·X_val,·y_train,·y_val·=·train_test_split(X_train,·y_train,·test_size=0.2)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 4s 0us/step
170508288/170498071 [==============================] - 4s 0us/step
```

```
#3. Check image shape
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
print(X_val.shape)
print(y_val.shape)
```

```
(40000, 32, 32, 3)
(40000, 10)
(10000, 32, 32, 3)
(10000, 10)
(10000, 32, 32, 3)
(10000, 10)
```

```
# ======= MODEL 1 ===========
```

```
from tensorflow.python.keras import Sequential
from tensorflow.python.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D

model_1 = Sequential()

model_1.add(Conv2D(32, (3, 3), activation='relu', padding ='same', input_shape=(32, 32
model_1.add(MaxPooling2D((2, 2)))
model_1.add(Conv2D(32, (3, 3), activation='relu', padding ='same'))
model_1.add(MaxPooling2D((2, 2)))
model_1.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_1.add(MaxPooling2D((2, 2)))
model_1.add(Conv2D(128, (3, 3), padding ='same', activation='relu'))
model_1.add(MaxPooling2D((2, 2)))

model_1.add(Flatten())
model_1.add(Dense(216, activation='relu'))
model_1.add(Dense(10, activation = "softmax"))

model_1.summary()

model_1.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['acc
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 128)         73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 216)               110808
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2170
=================================================================
Total params: 215,474
Trainable params: 215,474
Non-trainable params: 0
_________________________________________________________________
```

```
# Time how fast the model train

import time

start = time.time()
history = model_1.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1, val
end = time.time()

num_mins = (end-start)/60
print("Total training time: "  + str(num_mins) + " minutes.")
```

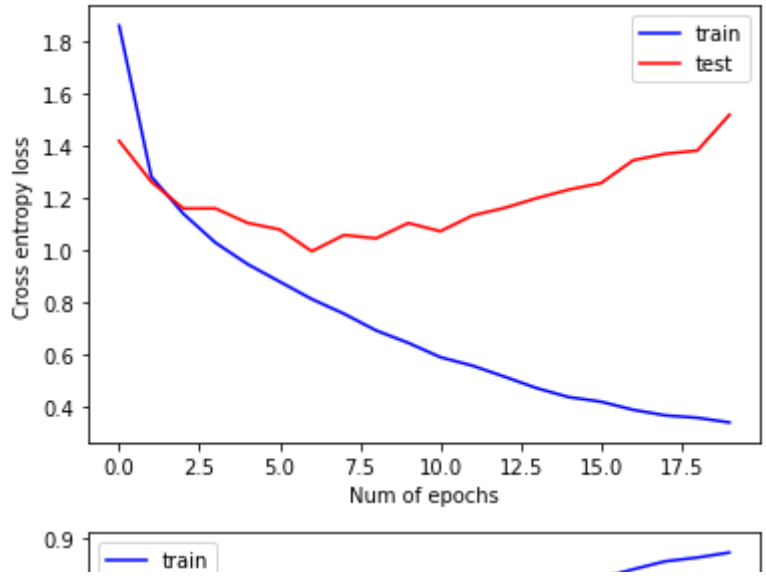
```
Epoch 1/20
625/625 [==============================] - 20s 16ms/step - loss: 1.8637 - accura
Epoch 2/20
625/625 [==============================] - 9s 15ms/step - loss: 1.2833 - accuracy
Epoch 3/20
625/625 [==============================] - 9s 15ms/step - loss: 1.1398 - accuracy
Epoch 4/20
625/625 [==============================] - 9s 15ms/step - loss: 1.0283 - accuracy
Epoch 5/20
625/625 [==============================] - 10s 15ms/step - loss: 0.9457 - accura
Epoch 6/20
625/625 [==============================] - 9s 15ms/step - loss: 0.8783 - accuracy
Epoch 7/20
625/625 [==============================] - 10s 16ms/step - loss: 0.8112 - accura
Epoch 8/20
625/625 [==============================] - 9s 15ms/step - loss: 0.7549 - accuracy
Epoch 9/20
625/625 [==============================] - 11s 18ms/step - loss: 0.6906 - accura
Epoch 10/20
625/625 [==============================] - 10s 16ms/step - loss: 0.6430 - accura
Epoch 11/20
625/625 [==============================] - 9s 15ms/step - loss: 0.5877 - accuracy
Epoch 12/20
625/625 [==============================] - 9s 15ms/step - loss: 0.5550 - accuracy
Epoch 13/20
625/625 [==============================] - 9s 15ms/step - loss: 0.5126 - accuracy
Epoch 14/20
625/625 [==============================] - 10s 16ms/step - loss: 0.4685 - accura
Epoch 15/20
625/625 [==============================] - 9s 15ms/step - loss: 0.4340 - accuracy
Epoch 16/20
625/625 [==============================] - 9s 15ms/step - loss: 0.4165 - accuracy
Epoch 17/20
625/625 [==============================] - 9s 15ms/step - loss: 0.3852 - accuracy
Epoch 18/20
625/625 [==============================] - 10s 16ms/step - loss: 0.3638 - accura
Epoch 19/20
625/625 [==============================] - 10s 16ms/step - loss: 0.3545 - accura
Epoch 20/20
625/625 [==============================] - 10s 16ms/step - loss: 0.3366 - accura
Total training time: 3.373894735177358 minutes.
```

```
# Loss and Accuracy
```

```
from sklearn.model_selection import KFold

scores = []
histories = []

loss, acc = model_1.evaluate(X_test, y_test, verbose =0)
print("Test loss: %.4f" % loss)
print("Test accuracy: %.2f" % (acc * 100.0))
scores.append(acc)
histories.append(history)
```

```
Test loss: 1.5470
Test accuracy: 66.04
```

```
# Plot loss function

from matplotlib import pyplot as plt

plt.plot(history.history["loss"], color = "blue", label = "train")
plt.plot(history.history["val_loss"], color = "red", label = "test")
plt.legend()
plt.ylabel("Cross entropy loss")
plt.xlabel("Num of epochs")
plt.show()

#1 Plot accuracy

plt.plot(history.history["accuracy"], color = "blue", label = "train")
plt.plot(history.history["val_accuracy"], color = "red", label = "test")
plt.legend()

plt.ylabel("Accuracy")
plt.xlabel("Num of epochs")
plt.show()
```




```
# ========= MODEL 2 ===========

from tensorflow.python.keras import Sequential
from tensorflow.python.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D

model_2 = Sequential()

model_2.add(Conv2D(32, (3, 3), activation='relu', padding ='same', input_shape=(32, 32
model_2.add(MaxPooling2D((2, 2)))
model_2.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_2.add(MaxPooling2D((2, 2)))
model_2.add(Conv2D(128, (3, 3), padding ='same', activation='relu'))
model_2.add(MaxPooling2D((2, 2)))

model_2.add(Flatten())
model_2.add(Dense(216, activation='relu'))
model_2.add(Dense(10, activation = "softmax"))

model_2.summary()

model_2.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['acc
```

```
Model: "sequential_1"
_________________________________________________________________
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, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
```

```
max_pooling2d_6 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 216)               442584
_________________________________________________________________
dense_3 (Dense)              (None, 10)                2170
=================================================================
Total params: 538,002
Trainable params: 538,002
Non-trainable params: 0
_________________________________________________________________
```

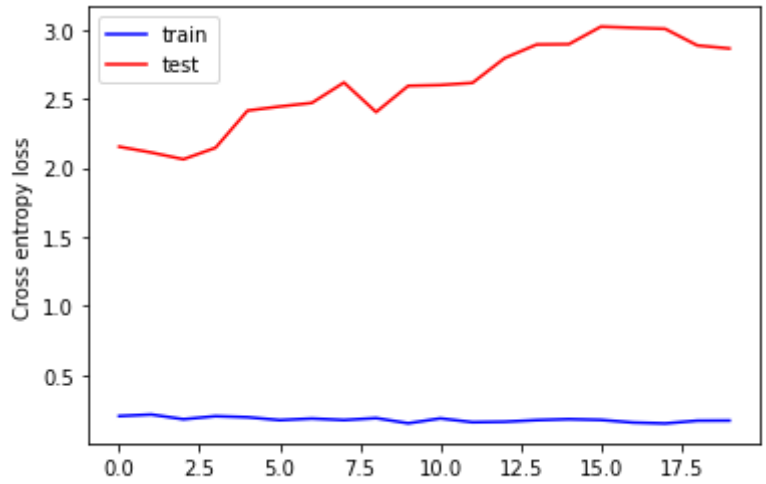
```
import time

start = time.time()
history = model_2.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1, val
end = time.time()

num_mins = (end-start)/60
print("Total training time: "  + str(num_mins) + " minutes.")
```

```
Epoch 1/20
625/625 [==============================] - 12s 18ms/step - loss: 0.2010 - accura
Epoch 2/20
625/625 [==============================] - 12s 19ms/step - loss: 0.2120 - accura
Epoch 3/20
625/625 [==============================] - 11s 18ms/step - loss: 0.1779 - accura
Epoch 4/20
625/625 [==============================] - 11s 18ms/step - loss: 0.2000 - accura
Epoch 5/20
625/625 [==============================] - 12s 19ms/step - loss: 0.1923 - accura
Epoch 6/20
625/625 [==============================] - 11s 17ms/step - loss: 0.1715 - accura
Epoch 7/20
625/625 [==============================] - 11s 18ms/step - loss: 0.1828 - accura
Epoch 8/20
625/625 [==============================] - 11s 18ms/step - loss: 0.1720 - accura
Epoch 9/20
625/625 [==============================] - 11s 18ms/step - loss: 0.1861 - accura
Epoch 10/20
625/625 [==============================] - 11s 17ms/step - loss: 0.1485 - accura
Epoch 11/20
625/625 [==============================] - 11s 18ms/step - loss: 0.1840 - accura
Epoch 12/20
625/625 [==============================] - 11s 18ms/step - loss: 0.1563 - accura
Epoch 13/20
625/625 [==============================] - 12s 19ms/step - loss: 0.1596 - accura
Epoch 14/20
625/625 [==============================] - 11s 18ms/step - loss: 0.1715 - accura
Epoch 15/20
625/625 [==============================] - 11s 18ms/step - loss: 0.1783 - accura
Epoch 16/20
625/625 [==============================] - 11s 17ms/step - loss: 0.1729 - accura
Epoch 17/20
```

```
625/625 [==============================] - 11s 18ms/step - loss: 0.1539 - accura
Epoch 18/20
625/625 [==============================] - 11s 18ms/step - loss: 0.1473 - accura
Epoch 19/20
625/625 [==============================] - 11s 18ms/step - loss: 0.1674 - accura
Epoch 20/20
625/625 [==============================] - 11s 17ms/step - loss: 0.1682 - accura
Total training time: 4.380449004968008 minutes.
```

```
# Loss and Accuracy

loss, acc = model_2.evaluate(X_test, y_test, verbose =0)
print("Test loss: %.4f" % loss)
print("Test accuracy: %.2f" % (acc * 100.0))

# Plot loss function

from matplotlib import pyplot as plt

plt.plot(history.history["loss"], color = "blue", label = "train")
plt.plot(history.history["val_loss"], color = "red", label = "test")
plt.legend()
plt.ylabel("Cross entropy loss")
plt.xlabel("Num of epochs")
plt.show()

#1 Plot accuracy

plt.plot(history.history["accuracy"], color = "blue", label = "train")
plt.plot(history.history["val_accuracy"], color = "red", label = "test")
plt.legend()

plt.ylabel("Accuracy")
plt.xlabel("Num of epochs")
plt.show()
```

```
Test loss: 2.8569
Test accuracy: 67.35
```




```
# ========= MODEL 3 ===========

model_3 = Sequential()

model_3.add(Conv2D(32, (3, 3), activation='relu', padding ='same', input_shape=(32, 32
model_3.add(MaxPooling2D((2, 2)))
model_3.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_3.add(MaxPooling2D((2, 2)))
model_3.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_3.add(MaxPooling2D((2, 2)))
model_3.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_3.add(MaxPooling2D((2, 2)))
model_3.add(Conv2D(128, (3, 3), padding ='same', activation='relu'))
model_3.add(MaxPooling2D((2, 2)))

model_3.add(Flatten())
model_3.add(Dense(216, activation='relu'))
model_3.add(Dense(10, activation = "softmax"))

model_3.summary()

model_3.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['acc
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
```

```
max_pooling2d_9 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 2, 2, 64)          0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 2, 2, 128)         73856
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 216)               27864
_________________________________________________________________
dense_5 (Dense)              (None, 10)                2170
=================================================================
Total params: 197,138
Trainable params: 197,138
Non-trainable params: 0
_________________________________________________________________
```

```
# Time how fast the model train

start = time.time()
history = model_3.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1, val
end = time.time()

num_mins = (end-start)/60
print("Total training time: "  + str(num_mins) + " minutes.")

# Loss and Accuracy

loss, acc = model_3.evaluate(X_test, y_test, verbose =0)
print("Test loss: %.4f" % loss)
print("Test accuracy: %.2f" % (acc * 100.0))

# Plot loss function

from matplotlib import pyplot as plt

plt.plot(history.history["loss"], color = "blue", label = "train")
plt.plot(history.history["val_loss"], color = "red", label = "test")
plt.legend()
plt.ylabel("Cross entropy loss")
plt.xlabel("Num of epochs")
plt.show()

#1 Plot accuracy

plt.plot(history.history["accuracy"], color = "blue", label = "train")
plt.plot(history.history["val_accuracy"], color = "red", label = "test")
plt.legend()
```

```
plt.ylabel("Accuracy")
plt.xlabel("Num of epochs")
plt.show()
```

```
Epoch 1/20
625/625 [==============================] - 12s 18ms/step - loss: 1.8078 - a
Epoch 2/20
625/625 [==============================] - 11s 18ms/step - loss: 1.3241 - a
Epoch 3/20
625/625 [==============================] - 11s 18ms/step - loss: 1.1605 - a
Epoch 4/20
625/625 [==============================] - 11s 18ms/step - loss: 1.0408 - a
Epoch 5/20
625/625 [==============================] - 11s 18ms/step - loss: 0.9464 - a
Epoch 6/20
625/625 [==============================] - 11s 18ms/step - loss: 0.8741 - a
Epoch 7/20
625/625 [==============================] - 12s 19ms/step - loss: 0.8209 - a
Epoch 8/20
625/625 [==============================] - 12s 20ms/step - loss: 0.7634 - a
Epoch 9/20
625/625 [==============================] - 12s 20ms/step - loss: 0.7144 - a
Epoch 10/20
625/625 [==============================] - 12s 19ms/step - loss: 0.6573 - a
Epoch 11/20
625/625 [==============================] - 12s 19ms/step - loss: 0.6178 - a
Epoch 12/20
```

```
# ========= MODEL 4 ===========

from keras import regularizers

model_4 = Sequential()

model_4.add(Conv2D(32, (3, 3), activation='relu', padding ='same', input_shape=(32, 32
model_4.add(MaxPooling2D((2, 2)))
model_4.add(Conv2D(64, (3, 3), padding ='same', activation='relu', kernel_regularizer=
model_4.add(MaxPooling2D((2, 2)))
model_4.add(Conv2D(64, (3, 3), padding ='same', activation='relu', kernel_regularizer=
model_4.add(MaxPooling2D((2, 2)))
model_4.add(Conv2D(64, (3, 3), padding ='same', activation='relu', kernel_regularizer=
model_4.add(MaxPooling2D((2, 2)))
model_4.add(Conv2D(128, (3, 3), padding ='same', activation='relu', kernel_regularizer
model_4.add(MaxPooling2D((2, 2)))

model_4.add(Flatten())
model_4.add(Dense(216, activation='relu', kernel_regularizer=regularizers.l2(0.00001))
model_4.add(Dense(10, activation = "softmax"))

model_4.summary()

model_4.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['acc
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 32, 32, 32)        896
```

```
max_pooling2d_12 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 2, 2, 64)          0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 2, 2, 128)         73856
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 216)               27864
_________________________________________________________________
dense_7 (Dense)              (None, 10)                2170
=================================================================
Total params: 197,138
Trainable params: 197,138
Non-trainable params: 0
_________________________________________________________________
```

```
# Time how fast the model train

start = time.time()
history = model_4.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1, val
end = time.time()

num_mins = (end-start)/60
print("Total training time: "  + str(num_mins) + " minutes.")

# Loss and Accuracy

loss, acc = model_4.evaluate(X_test, y_test, verbose =0)
print("Test loss: %.4f" % loss)
print("Test accuracy: %.2f" % (acc * 100.0))

# Plot loss function

from matplotlib import pyplot as plt

plt.plot(history.history["loss"], color = "blue", label = "train")
plt.plot(history.history["val_loss"], color = "red", label = "test")
plt.legend()
plt.ylabel("Cross entropy loss")
plt.xlabel("Num of epochs")
```

```
plt.xlabel("Num of epochs")
plt.show()

#1 Plot accuracy

plt.plot(history.history["accuracy"], color = "blue", label = "train")
plt.plot(history.history["val_accuracy"], color = "red", label = "test")
plt.legend()

plt.ylabel("Accuracy")
plt.xlabel("Num of epochs")
plt.show()
```

```
Epoch 1/20
625/625 [==============================] - 15s 21ms/step - loss: 1.6141 - a
Epoch 2/20
625/625 [==============================] - 12s 19ms/step - loss: 1.2214 - a
Epoch 3/20
625/625 [==============================] - 12s 19ms/step - loss: 1.0559 - a
Epoch 4/20
625/625 [==============================] - 12s 19ms/step - loss: 0.9579 - a
Epoch 5/20
625/625 [==============================] - 11s 18ms/step - loss: 0.8755 - a
Epoch 6/20
625/625 [==============================] - 11s 18ms/step - loss: 0.8092 - a
Epoch 7/20
625/625 [==============================] - 11s 18ms/step - loss: 0.7528 - a
Epoch 8/20
625/625 [==============================] - 12s 19ms/step - loss: 0.7023 - a
Epoch 9/20
625/625 [==============================] - 12s 20ms/step - loss: 0.6555 - a
Epoch 10/20
625/625 [==============================] - 12s 18ms/step - loss: 0.6236 - a
Epoch 11/20
625/625 [==============================] - 12s 19ms/step - loss: 0.5808 - a
Epoch 12/20
625/625 [==============================] - 12s 19ms/step - loss: 0.5418 - a
Epoch 13/20
625/625 [==============================] - 12s 19ms/step - loss: 0.5334 - a
Epoch 14/20
625/625 [==============================] - 12s 19ms/step - loss: 0.4888 - a
Epoch 15/20
625/625 [==============================] - 12s 20ms/step - loss: 0.4549 - a
Epoch 16/20
```

```
# ========= MODEL 5 ===========

from keras import regularizers
from tensorflow.python.keras.layers.core import Dropout

model_5 = Sequential()

model_5.add(Conv2D(32, (3, 3), activation='relu', padding ='same', input_shape=(32, 32
model_5.add(MaxPooling2D((2, 2)))
model_5.add(Dropout(0.1))
model_5.add(Conv2D(64, (3, 3), padding ='same', activation='relu', kernel_regularizer=
model_5.add(MaxPooling2D((2, 2)))
model_5.add(Dropout(0.1))
model_5.add(Conv2D(64, (3, 3), padding ='same', activation='relu', kernel_regularizer=
model_5.add(MaxPooling2D((2, 2)))
model_5.add(Dropout(0.1))
model_5.add(Conv2D(64, (3, 3), padding ='same', activation='relu', kernel_regularizer=
model_5.add(MaxPooling2D((2, 2)))
model_5.add(Dropout(0.1))
model_5.add(Conv2D(128, (3, 3), padding ='same', activation='relu', kernel_regularizer
model_5.add(MaxPooling2D((2, 2)))
model_5.add(Dropout(0.1))
```

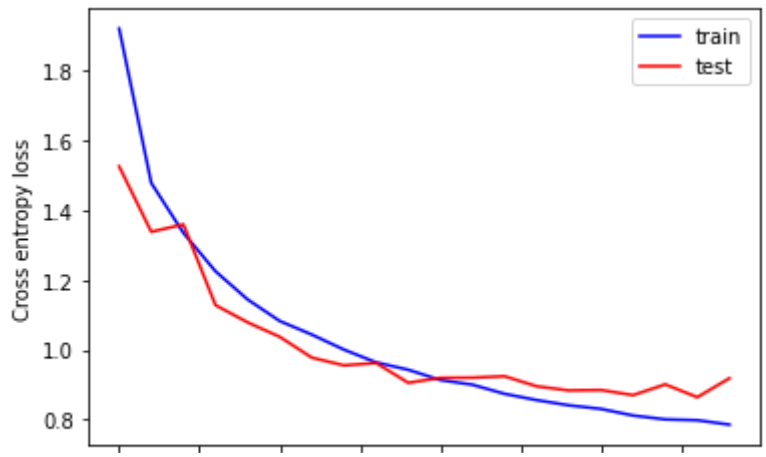
```
model_5.add(Flatten())
model_5.add(Dense(216, activation='relu', kernel_regularizer=regularizers.l2(0.00001)))
model_5.add(Dropout(0.1))
model_5.add(Dense(10, activation = "softmax"))

model_5.summary()

model_5.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['acc
```

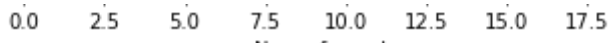
```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 4, 4, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 2, 2, 64)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 2, 64)          0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 2, 2, 128)         73856
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 1, 1, 128)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 216)               27864
_________________________________________________________________
dropout_5 (Dropout)          (None, 216)               0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                2170
=================================================================
Total params: 197,138
Trainable params: 197,138
```

```
Non-trainable params: 0
_________________________________________________________________
```

```
# Time how fast the model train

start = time.time()
history = model_5.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1, val
end = time.time()

num_mins = (end-start)/60
print("Total training time: "  + str(num_mins) + " minutes.")

# Loss and Accuracy

loss, acc = model_5.evaluate(X_test, y_test, verbose =0)
print("Test loss: %.4f" % loss)
print("Test accuracy: %.2f" % (acc * 100.0))

# Plot loss function

from matplotlib import pyplot as plt

plt.plot(history.history["loss"], color = "blue", label = "train")
plt.plot(history.history["val_loss"], color = "red", label = "test")
plt.legend()
plt.ylabel("Cross entropy loss")
plt.xlabel("Num of epochs")
plt.show()

#1 Plot accuracy

plt.plot(history.history["accuracy"], color = "blue", label = "train")
plt.plot(history.history["val_accuracy"], color = "red", label = "test")
plt.legend()

plt.ylabel("Accuracy")
plt.xlabel("Num of epochs")
plt.show()
```

```
Epoch 1/20
625/625 [==============================] - 16s 21ms/step - loss: 1.9195 - a
Epoch 2/20
625/625 [==============================] - 13s 21ms/step - loss: 1.4776 - a
Epoch 3/20
625/625 [==============================] - 13s 20ms/step - loss: 1.3347 - a
Epoch 4/20
625/625 [==============================] - 13s 20ms/step - loss: 1.2248 - a
Epoch 5/20
625/625 [==============================] - 13s 21ms/step - loss: 1.1442 - a
Epoch 6/20
625/625 [==============================] - 13s 22ms/step - loss: 1.0825 - a
Epoch 7/20
625/625 [==============================] - 15s 25ms/step - loss: 1.0436 - a
Epoch 8/20
625/625 [==============================] - 13s 21ms/step - loss: 1.0006 - a
Epoch 9/20
625/625 [==============================] - 14s 22ms/step - loss: 0.9641 - a
Epoch 10/20
625/625 [==============================] - 13s 21ms/step - loss: 0.9431 - a
Epoch 11/20
625/625 [==============================] - 13s 20ms/step - loss: 0.9136 - a
Epoch 12/20
625/625 [==============================] - 13s 21ms/step - loss: 0.9003 - a
Epoch 13/20
625/625 [==============================] - 13s 20ms/step - loss: 0.8742 - a
Epoch 14/20
625/625 [==============================] - 13s 20ms/step - loss: 0.8562 - a
Epoch 15/20
625/625 [==============================] - 13s 21ms/step - loss: 0.8413 - a
Epoch 16/20
625/625 [==============================] - 15s 23ms/step - loss: 0.8310 - a
Epoch 17/20
625/625 [==============================] - 13s 21ms/step - loss: 0.8121 - a
Epoch 18/20
625/625 [==============================] - 13s 20ms/step - loss: 0.8011 - a
Epoch 19/20
625/625 [==============================] - 13s 20ms/step - loss: 0.7986 - a
Epoch 20/20
625/625 [==============================] - 13s 20ms/step - loss: 0.7860 - a
Total training time: 5.420197983582814 minutes.
Test loss: 0.9254
Test accuracy: 69.62
```






```
# ========= MODEL 6 ===========

from keras import regularizers
from tensorflow.keras.layers import BatchNormalization
from tensorflow.python.keras import Sequential
from tensorflow.python.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D

model_6 = Sequential()

model_6.add(Conv2D(32, (3, 3), activation='relu', padding ='same', input_shape=(32, 32
model_6.add(MaxPooling2D((2, 2)))
model_6.add(BatchNormalization())
model_6.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_6.add(MaxPooling2D((2, 2)))
model_6.add(BatchNormalization())
model_6.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_6.add(MaxPooling2D((2, 2)))
model_6.add(BatchNormalization())
model_6.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_6.add(MaxPooling2D((2, 2)))
model_6.add(BatchNormalization())
model_6.add(Conv2D(128, (3, 3), padding ='same', activation='relu'))
model_6.add(MaxPooling2D((2, 2)))
model_6.add(BatchNormalization())

model_6.add(Flatten())
model_6.add(Dense(216, activation='relu'))
model_6.add(BatchNormalization())
model_6.add(Dense(10, activation = "softmax"))

model_6.summary()

model_6.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['acc
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_27 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
module_wrapper_2 (ModuleWrap (None, 16, 16, 32)        128
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
module_wrapper_3 (ModuleWrap (None, 8, 8, 64)          256
```

```
conv2d_29 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 4, 4, 64)          0
_________________________________________________________________
module_wrapper_4 (ModuleWrap (None, 4, 4, 64)          256
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_30 (MaxPooling (None, 2, 2, 64)          0
_________________________________________________________________
module_wrapper_5 (ModuleWrap (None, 2, 2, 64)          256
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 2, 2, 128)         73856
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
module_wrapper_6 (ModuleWrap (None, 1, 1, 128)         512
_________________________________________________________________
flatten_6 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_12 (Dense)             (None, 216)               27864
_________________________________________________________________
module_wrapper_7 (ModuleWrap (None, 216)               864
_________________________________________________________________
dense_13 (Dense)             (None, 10)                2170
=================================================================
Total params: 199,410
Trainable params: 198,274
Non-trainable params: 1,136
_________________________________________________________________
```

```
# Time how fast the model train

start = time.time()
history = model_6.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1, val
end = time.time()

num_mins = (end-start)/60
print("Total training time: "  + str(num_mins) + " minutes.")

# Loss and Accuracy

loss, acc = model_6.evaluate(X_test, y_test, verbose =0)
print("Test loss: %.4f" % loss)
print("Test accuracy: %.2f" % (acc * 100.0))

# Plot loss function

from matplotlib import pyplot as plt

plt.plot(history.history["loss"], color = "blue", label = "train")
plt.plot(history.history["val_loss"], color = "red", label = "test")
```

```
plt.legend()
plt.ylabel("Cross entropy loss")
plt.xlabel("Num of epochs")
plt.show()

#1 Plot accuracy

plt.plot(history.history["accuracy"], color = "blue", label = "train")
plt.plot(history.history["val_accuracy"], color = "red", label = "test")
plt.legend()

plt.ylabel("Accuracy")
plt.xlabel("Num of epochs")
plt.show()
```

```
Epoch 1/20
625/625 [==============================] - 15s 21ms/step - loss: 1.4077 - a
Epoch 2/20
625/625 [==============================] - 10s 16ms/step - loss: 0.9841 - a
Epoch 3/20
625/625 [==============================] - 11s 17ms/step - loss: 0.8070 - a
Epoch 4/20
625/625 [==============================] - 12s 20ms/step - loss: 0.6817 - a
Epoch 5/20
625/625 [==============================] - 13s 21ms/step - loss: 0.5856 - a
Epoch 6/20
625/625 [==============================] - 13s 20ms/step - loss: 0.5025 - a
Epoch 7/20
625/625 [==============================] - 13s 21ms/step - loss: 0.4294 - a
Epoch 8/20
625/625 [==============================] - 13s 20ms/step - loss: 0.3640 - a
Epoch 9/20
625/625 [==============================] - 13s 20ms/step - loss: 0.3089 - a
Epoch 10/20
625/625 [==============================] - 14s 22ms/step - loss: 0.2655 - a
Epoch 11/20
625/625 [==============================] - 13s 21ms/step - loss: 0.2348 - a
Epoch 12/20
625/625 [==============================] - 14s 22ms/step - loss: 0.2029 - a
Epoch 13/20
625/625 [==============================] - 15s 23ms/step - loss: 0.1791 - a
Epoch 14/20
625/625 [==============================] - 13s 21ms/step - loss: 0.1666 - a
Epoch 15/20
625/625 [==============================] - 13s 21ms/step - loss: 0.1567 - a
Epoch 16/20
625/625 [==============================] - 13s 21ms/step - loss: 0.1381 - a
Epoch 17/20
625/625 [==============================] - 13s 21ms/step - loss: 0.1362 - a
```

```
# ========= MODEL 7 ===========

from keras import regularizers
from tensorflow.keras.layers import BatchNormalization, Dropout

model_7 = Sequential()

model_7.add(Conv2D(32, (3, 3), activation='relu', padding ='same', input_shape=(32, 32
model_7.add(MaxPooling2D((2, 2)))
model_7.add(BatchNormalization())
model_7.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_7.add(MaxPooling2D((2, 2)))
model_7.add(Dropout(0.1))
model_7.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_7.add(MaxPooling2D((2, 2)))
model_7.add(BatchNormalization())
model_7.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_7.add(MaxPooling2D((2, 2)))
model_7.add(Dropout(0.1))
```

```
model_7.add(Conv2D(128, (3, 3), padding ='same', activation='relu'))
model_7.add(MaxPooling2D((2, 2)))
model_7.add(BatchNormalization())

model_7.add(Flatten())
model_7.add(Dense(216, activation='relu'))
model_7.add(Dropout(0.1))
model_7.add(Dense(10, activation = "softmax"))

model_7.summary()

model_7.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['acc
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_32 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
module_wrapper_8 (ModuleWrap (None, 16, 16, 32)        128
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
module_wrapper_9 (ModuleWrap (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_34 (MaxPooling (None, 4, 4, 64)          0
_________________________________________________________________
module_wrapper_10 (ModuleWra (None, 4, 4, 64)          256
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_35 (MaxPooling (None, 2, 2, 64)          0
_________________________________________________________________
module_wrapper_11 (ModuleWra (None, 2, 2, 64)          0
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 2, 2, 128)         73856
_________________________________________________________________
max_pooling2d_36 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
module_wrapper_12 (ModuleWra (None, 1, 1, 128)         512
_________________________________________________________________
flatten_7 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_14 (Dense)             (None, 216)               27864
_________________________________________________________________
module_wrapper_13 (ModuleWra (None, 216)               0
_________________________________________________________________
dense_15 (Dense)             (None, 10)                2170
=================================================================
```

```
Total params: 198,034
Trainable params: 197,586
Non-trainable params: 448
_________________________________________________________________
```

```
# Time how fast the model train

start = time.time()
history = model_7.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1, val
end = time.time()

num_mins = (end-start)/60
print("Total training time: "  + str(num_mins) + " minutes.")

# Loss and Accuracy

loss, acc = model_7.evaluate(X_test, y_test, verbose =0)
print("Test loss: %.4f" % loss)
print("Test accuracy: %.2f" % (acc * 100.0))

# Plot loss function

from matplotlib import pyplot as plt

plt.plot(history.history["loss"], color = "blue", label = "train")
plt.plot(history.history["val_loss"], color = "red", label = "test")
plt.legend()
plt.ylabel("Cross entropy loss")
plt.xlabel("Num of epochs")
plt.show()

#1 Plot accuracy

plt.plot(history.history["accuracy"], color = "blue", label = "train")
plt.plot(history.history["val_accuracy"], color = "red", label = "test")
plt.legend()

plt.ylabel("Accuracy")
plt.xlabel("Num of epochs")
plt.show()
```

```
Epoch 1/20
625/625 [==============================] - 14s 20ms/step - loss: 1.4105 - a
Epoch 2/20
625/625 [==============================] - 13s 20ms/step - loss: 1.0447 - a
Epoch 3/20
625/625 [==============================] - 12s 19ms/step - loss: 0.8971 - a
Epoch 4/20
625/625 [==============================] - 13s 20ms/step - loss: 0.8017 - a
Epoch 5/20
625/625 [==============================] - 12s 20ms/step - loss: 0.7286 - a
Epoch 6/20
625/625 [==============================] - 12s 19ms/step - loss: 0.6738 - a
Epoch 7/20
625/625 [==============================] - 12s 19ms/step - loss: 0.6238 - a
Epoch 8/20
625/625 [==============================] - 12s 20ms/step - loss: 0.5774 - a
Epoch 9/20
625/625 [==============================] - 13s 20ms/step - loss: 0.5414 - a
Epoch 10/20
625/625 [==============================] - 12s 20ms/step - loss: 0.5086 - a
Epoch 11/20
625/625 [==============================] - 12s 19ms/step - loss: 0.4754 - a
Epoch 12/20
625/625 [==============================] - 14s 23ms/step - loss: 0.4512 - a
Epoch 13/20
625/625 [==============================] - 13s 21ms/step - loss: 0.4217 - a
Epoch 14/20
625/625 [==============================] - 13s 21ms/step - loss: 0.4029 - a
Epoch 15/20
625/625 [==============================] - 14s 22ms/step - loss: 0.3876 - a
Epoch 16/20
625/625 [==============================] - 13s 21ms/step - loss: 0.3639 - a
Epoch 17/20
625/625 [==============================] - 12s 20ms/step - loss: 0.3517 - a
Epoch 18/20
625/625 [==============================] - 12s 19ms/step - loss: 0.3316 - a
Epoch 19/20
625/625 [==============================] - 12s 19ms/step - loss: 0.3170 - a
Epoch 20/20
625/625 [==============================] - 12s 19ms/step - loss: 0.3081 - a
Total training time: 4.390576024850209 minutes.
Test loss: 0.8388
Test accuracy: 75.48
```


```
# ========= MODEL 8 ===========

from keras import regularizers
from tensorflow.python.keras.layers.core import Dropout

model_8 = Sequential()

model_8.add(Conv2D(32, (3, 3), activation='relu', padding ='same', input_shape=(32, 32
```

```
model_8.add(MaxPooling2D((2, 2)))
model_8.add(Dropout(0.1))
model_8.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_8.add(MaxPooling2D((2, 2)))
model_8.add(Dropout(0.1))
model_8.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_8.add(MaxPooling2D((2, 2)))
model_8.add(Dropout(0.1))
model_8.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_8.add(MaxPooling2D((2, 2)))
model_8.add(Dropout(0.1))
model_8.add(Conv2D(128, (3, 3), padding ='same', activation='relu'))
model_8.add(MaxPooling2D((2, 2)))
model_8.add(Dropout(0.1))

model_8.add(Flatten())
model_8.add(Dense(216, activation='relu'))
model_8.add(Dropout(0.1))
model_8.add(Dense(10, activation = "softmax"))

model_8.summary()

model_8.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['acc
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_37 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_37 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_38 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
dropout_7 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_39 (MaxPooling (None, 4, 4, 64)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_40 (MaxPooling (None, 2, 2, 64)          0
_________________________________________________________________
dropout_9 (Dropout)          (None, 2, 2, 64)          0
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 2, 2, 128)         73856
_________________________________________________________________
```

```
max_pooling2d_41 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
dropout_10 (Dropout)         (None, 1, 1, 128)         0
_________________________________________________________________
flatten_8 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_16 (Dense)             (None, 216)               27864
_________________________________________________________________
dropout_11 (Dropout)         (None, 216)               0
_________________________________________________________________
dense_17 (Dense)             (None, 10)                2170
=================================================================
Total params: 197,138
Trainable params: 197,138
Non-trainable params: 0
_________________________________________________________________
```

```
# Time how fast the model train

start = time.time()
history = model_8.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1, val
end = time.time()

num_mins = (end-start)/60
print("Total training time: "  + str(num_mins) + " minutes.")

# Loss and Accuracy

loss, acc = model_8.evaluate(X_test, y_test, verbose =0)
print("Test loss: %.4f" % loss)
print("Test accuracy: %.2f" % (acc * 100.0))

# Plot loss function

from matplotlib import pyplot as plt

plt.plot(history.history["loss"], color = "blue", label = "train")
plt.plot(history.history["val_loss"], color = "red", label = "test")
plt.legend()
plt.ylabel("Cross entropy loss")
plt.xlabel("Num of epochs")
plt.show()

#1 Plot accuracy

plt.plot(history.history["accuracy"], color = "blue", label = "train")
plt.plot(history.history["val_accuracy"], color = "red", label = "test")
plt.legend()

plt.ylabel("Accuracy")
plt.xlabel("Num of epochs")
plt.show()
```

```
Epoch 1/20
625/625 [==============================] - 14s 21ms/step - loss: 1.8781 - a
Epoch 2/20
625/625 [==============================] - 12s 19ms/step - loss: 1.4723 - a
Epoch 3/20
625/625 [==============================] - 12s 20ms/step - loss: 1.3096 - a
Epoch 4/20
625/625 [==============================] - 13s 20ms/step - loss: 1.2191 - a
Epoch 5/20
625/625 [==============================] - 12s 19ms/step - loss: 1.1329 - a
Epoch 6/20
625/625 [==============================] - 12s 20ms/step - loss: 1.0713 - a
Epoch 7/20
625/625 [==============================] - 12s 20ms/step - loss: 1.0183 - a
Epoch 8/20
625/625 [==============================] - 13s 21ms/step - loss: 0.9899 - a
Epoch 9/20
625/625 [==============================] - 12s 19ms/step - loss: 0.9582 - a
Epoch 10/20
```

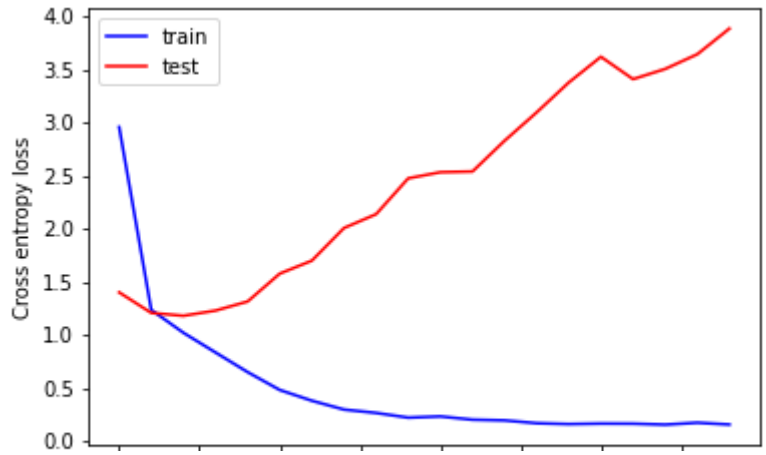
```
# ======= MODEL 9 ===========

from tensorflow.python.keras import Sequential
from tensorflow.python.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D

model_9 = Sequential()

model_9.add(Conv2D(32, (3, 3), activation='relu', padding ='same', input_shape=(32, 32
model_9.add(MaxPooling2D((2, 2)))
model_9.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_9.add(MaxPooling2D((2, 2)))

model_9.add(Flatten())
model_9.add(Dense(512, activation='relu'))
model_9.add(Dense(10, activation = "softmax"))

model_9.summary()

model_9.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['acc
```

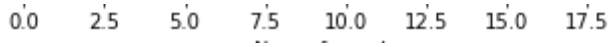
```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_42 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_42 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_43 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_43 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
flatten_9 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_18 (Dense)             (None, 512)               2097664
```

```
_________________________________________________________________
dense_19 (Dense)             (None, 10)                5130      
=================================================================
Total params: 2,122,186
Trainable params: 2,122,186
Non-trainable params: 0
_________________________________________________________________
```

```
# Time how fast the model train

start = time.time()
history = model_9.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1, val
end = time.time()

num_mins = (end-start)/60
print("Total training time: "  + str(num_mins) + " minutes.")

# Loss and Accuracy

loss, acc = model_9.evaluate(X_test, y_test, verbose =0)
print("Test loss: %.4f" % loss)
print("Test accuracy: %.2f" % (acc * 100.0))

# Plot loss function

from matplotlib import pyplot as plt

plt.plot(history.history["loss"], color = "blue", label = "train")
plt.plot(history.history["val_loss"], color = "red", label = "test")
plt.legend()
plt.ylabel("Cross entropy loss")
plt.xlabel("Num of epochs")
plt.show()

#1 Plot accuracy

plt.plot(history.history["accuracy"], color = "blue", label = "train")
plt.plot(history.history["val_accuracy"], color = "red", label = "test")
plt.legend()

plt.ylabel("Accuracy")
plt.xlabel("Num of epochs")
plt.show()
```

```
Epoch 1/20
625/625 [==============================] - 10s 15ms/step - loss: 2.9560 - a
Epoch 2/20
625/625 [==============================] - 9s 14ms/step - loss: 1.2343 - ac
Epoch 3/20
625/625 [==============================] - 9s 14ms/step - loss: 1.0190 - ac
Epoch 4/20
625/625 [==============================] - 9s 14ms/step - loss: 0.8333 - ac
Epoch 5/20
625/625 [==============================] - 8s 13ms/step - loss: 0.6485 - ac
Epoch 6/20
625/625 [==============================] - 8s 13ms/step - loss: 0.4802 - ac
Epoch 7/20
625/625 [==============================] - 8s 13ms/step - loss: 0.3790 - ac
Epoch 8/20
625/625 [==============================] - 9s 14ms/step - loss: 0.2958 - ac
Epoch 9/20
625/625 [==============================] - 8s 13ms/step - loss: 0.2640 - ac
Epoch 10/20
625/625 [==============================] - 9s 14ms/step - loss: 0.2197 - ac
Epoch 11/20
625/625 [==============================] - 8s 13ms/step - loss: 0.2310 - ac
Epoch 12/20
625/625 [==============================] - 9s 14ms/step - loss: 0.2004 - ac
Epoch 13/20
625/625 [==============================] - 9s 14ms/step - loss: 0.1936 - ac
Epoch 14/20
625/625 [==============================] - 8s 13ms/step - loss: 0.1671 - ac
Epoch 15/20
625/625 [==============================] - 8s 13ms/step - loss: 0.1591 - ac
Epoch 16/20
625/625 [==============================] - 9s 14ms/step - loss: 0.1638 - ac
Epoch 17/20
625/625 [==============================] - 9s 14ms/step - loss: 0.1628 - ac
Epoch 18/20
625/625 [==============================] - 8s 13ms/step - loss: 0.1538 - ac
Epoch 19/20
625/625 [==============================] - 9s 14ms/step - loss: 0.1719 - ac
Epoch 20/20
625/625 [==============================] - 8s 13ms/step - loss: 0.1548 - ac
Total training time: 2.898869283994039 minutes.
Test loss: 3.9393
Test accuracy: 59.14
```





Top-Performing Model: Model_7

```
# Import dataset
from keras.datasets import cifar10
import numpy as np
from sklearn.model_selection import train_test_split

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

from tensorflow.keras.utils import to_categorical

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)


# Time how fast the model train
import time
start = time.time()
history = model_7.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1)
end = time.time()

num_mins = (end-start)/60
print("Total training time: "  + str(num_mins) + " minutes.")

# Loss and Accuracy

loss, acc = model_7.evaluate(X_test, y_test, verbose =0)
print("Test loss: %.4f" % loss)
print("Test accuracy: %.2f" % (acc * 100.0))
```

```
Epoch 1/20
782/782 [==============================] - 14s 17ms/step - loss: 0.4061 - accura
Epoch 2/20
782/782 [==============================] - 12s 15ms/step - loss: 0.3666 - accura
Epoch 3/20
782/782 [==============================] - 12s 16ms/step - loss: 0.3542 - accura
Epoch 4/20
782/782 [==============================] - 12s 16ms/step - loss: 0.3372 - accura
Epoch 5/20
782/782 [==============================] - 12s 16ms/step - loss: 0.3236 - accura
Epoch 6/20
782/782 [==============================] - 12s 16ms/step - loss: 0.3088 - accura
```