CODING: Impact of Training Data Amount

```
# 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)


# ======= MODEL 1 =========== 100% Training Data ====== No Regularization ======

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(512, 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, 512)               262656
_________________________________________________________________
dense_1 (Dense)              (None, 10)                5130
=================================================================
Total params: 370,282
Trainable params: 370,282
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.")

# Loss and Accuracy

loss, acc = model_1.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 [==============================] - 21s 16ms/step - loss: 1.7180 - a
Epoch 2/20
625/625 [==============================] - 11s 18ms/step - loss: 1.2415 - a
Epoch 3/20
625/625 [==============================] - 11s 18ms/step - loss: 1.0995 - a
Epoch 4/20
625/625 [==============================] - 11s 17ms/step - loss: 0.9793 - a
Epoch 5/20
625/625 [==============================] - 11s 17ms/step - loss: 0.8879 - a
Epoch 6/20
625/625 [==============================] - 10s 17ms/step - loss: 0.8159 - a
Epoch 7/20
625/625 [==============================] - 11s 17ms/step - loss: 0.7533 - a
Epoch 8/20
625/625 [==============================] - 10s 16ms/step - loss: 0.6888 - a
Epoch 9/20
625/625 [==============================] - 10s 17ms/step - loss: 0.6280 - a
Epoch 10/20
625/625 [==============================] - 10s 16ms/step - loss: 0.5721 - a
Epoch 11/20
625/625 [==============================] - 10s 17ms/step - loss: 0.5326 - a
Epoch 12/20
625/625 [==============================] - 11s 18ms/step - loss: 0.4738 - a
Epoch 13/20
625/625 [==============================] - 11s 18ms/step - loss: 0.4432 - a
Epoch 14/20
625/625 [==============================] - 11s 18ms/step - loss: 0.4081 - a
Epoch 15/20
625/625 [==============================] - 12s 19ms/step - loss: 0.3702 - a
Epoch 16/20
625/625 [==============================] - 10s 17ms/step - loss: 0.3541 - a
Epoch 17/20
625/625 [==============================] - 11s 18ms/step - loss: 0.3353 - a
Epoch 18/20
```

```
# ======= MODEL 2 =========== 100% Training Data ====== With Regularization ======

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

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(Dropout(0.1))
model_2.add(Conv2D(32, (3, 3), activation='relu', padding ='same'))
model_2.add(MaxPooling2D((2, 2)))
model_2.add(Dropout(0.1))
model_2.add(Conv2D(64, (3, 3), padding ='same', activation='relu'))
model_2.add(MaxPooling2D((2, 2)))
model_2.add(Dropout(0.1))
model_2.add(Conv2D(128, (3, 3), padding ='same', activation='relu'))
```

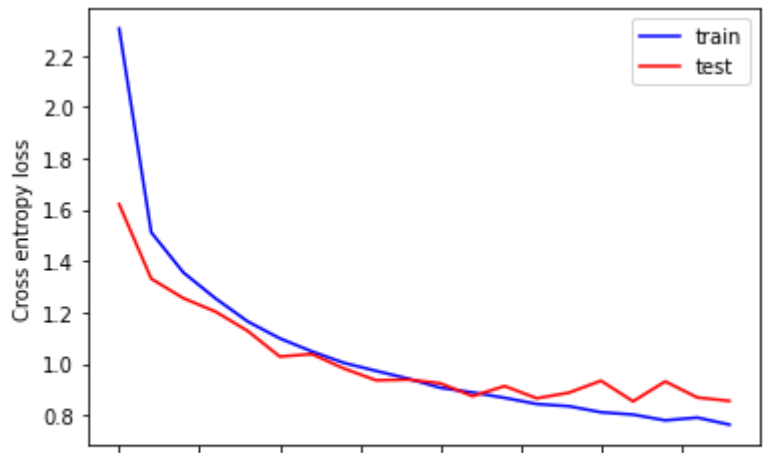
```
model_2.add(MaxPooling2D((2, 2)))
model_2.add(Dropout(0.1))

model_2.add(Flatten())
model_2.add(Dense(512, activation='relu'))
model_2.add(Dropout(0.1))
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         
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 16, 16, 32)        9248      
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 32)          0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 32)          0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 64)          18496     
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 4, 4, 64)          0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 64)          0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 4, 128)         73856     
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 2, 2, 128)         0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 2, 128)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656    
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130      
=================================================================
Total params: 370,282
Trainable params: 370,282
Non-trainable params: 0
_________________________________________________________________
```

```
# Time how fast the model train
```

```
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.")

# 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()
```

```
Epoch 1/20
625/625 [==============================] - 14s 20ms/step - loss: 2.3047 - a
Epoch 2/20
625/625 [==============================] - 13s 21ms/step - loss: 1.5116 - a
Epoch 3/20
625/625 [==============================] - 12s 18ms/step - loss: 1.3555 - a
Epoch 4/20
625/625 [==============================] - 12s 20ms/step - loss: 1.2547 - a
Epoch 5/20
625/625 [==============================] - 12s 19ms/step - loss: 1.1651 - a
Epoch 6/20
625/625 [==============================] - 14s 22ms/step - loss: 1.0991 - a
Epoch 7/20
625/625 [==============================] - 11s 18ms/step - loss: 1.0482 - a
Epoch 8/20
625/625 [==============================] - 11s 18ms/step - loss: 1.0037 - a
Epoch 9/20
625/625 [==============================] - 11s 17ms/step - loss: 0.9725 - a
Epoch 10/20
625/625 [==============================] - 11s 18ms/step - loss: 0.9426 - a
Epoch 11/20
625/625 [==============================] - 11s 17ms/step - loss: 0.9069 - a
Epoch 12/20
625/625 [==============================] - 11s 17ms/step - loss: 0.8887 - a
Epoch 13/20
625/625 [==============================] - 11s 18ms/step - loss: 0.8678 - a
Epoch 14/20
625/625 [==============================] - 11s 18ms/step - loss: 0.8437 - a
Epoch 15/20
625/625 [==============================] - 11s 17ms/step - loss: 0.8352 - a
Epoch 16/20
625/625 [==============================] - 11s 18ms/step - loss: 0.8117 - a
Epoch 17/20
625/625 [==============================] - 11s 17ms/step - loss: 0.8026 - a
Epoch 18/20
625/625 [==============================] - 11s 18ms/step - loss: 0.7801 - a
Epoch 19/20
625/625 [==============================] - 11s 17ms/step - loss: 0.7909 - a
Epoch 20/20
625/625 [==============================] - 11s 17ms/step - loss: 0.7635 - a
Total training time: 4.406443663438162 minutes.
Test loss: 0.8474
Test accuracy: 71.64
```



75% Training Data

```
# ======= 75% Training Data ======


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_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2)
X_train, X_test75, y_train, y_test75 = train_test_split(X_train, y_train, train_size=(
```

MODEL 3 : 75% Training Data | No Regularization

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

# Time how fast the model train

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.")

# Loss and Accuracy

loss, acc = model_1.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
469/469 [==============================] - 10s 21ms/step - loss: 0.7688 - a
Epoch 2/20
469/469 [==============================] - 8s 17ms/step - loss: 0.4223 - ac
Epoch 3/20
469/469 [==============================] - 7s 15ms/step - loss: 0.2999 - ac
Epoch 4/20
469/469 [==============================] - 7s 15ms/step - loss: 0.2471 - ac
Epoch 5/20
469/469 [==============================] - 7s 15ms/step - loss: 0.2513 - ac
Epoch 6/20
469/469 [==============================] - 7s 15ms/step - loss: 0.2573 - ac
Epoch 7/20
469/469 [==============================] - 7s 16ms/step - loss: 0.2128 - ac
Epoch 8/20
469/469 [==============================] - 7s 15ms/step - loss: 0.2298 - ac
Epoch 9/20
469/469 [==============================] - 7s 16ms/step - loss: 0.2033 - ac
Epoch 10/20
469/469 [==============================] - 7s 14ms/step - loss: 0.2455 - ac
Epoch 11/20
469/469 [==============================] - 7s 16ms/step - loss: 0.2039 - ac
Epoch 12/20
469/469 [==============================] - 7s 14ms/step - loss: 0.2031 - ac
Epoch 13/20
469/469 [==============================] - 7s 15ms/step - loss: 0.2260 - ac
Epoch 14/20
469/469 [==============================] - 7s 16ms/step - loss: 0.1910 - ac
Epoch 15/20
469/469 [==============================] - 7s 14ms/step - loss: 0.1990 - ac
Epoch 16/20
```

MODEL 4 : 75% Training Data | With Regularization

```
469/469 [==============================] - 7s 15ms/step - loss: 0.2267 - ac
```

```
# Time how fast the model train

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.")

# 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()
```

```
Epoch 1/20
469/469 [==============================] - 8s 16ms/step - loss: 0.7666 - ac
Epoch 2/20
469/469 [==============================] - 8s 17ms/step - loss: 0.7213 - ac
Epoch 3/20
469/469 [==============================] - 8s 17ms/step - loss: 0.6854 - ac
Epoch 4/20
469/469 [==============================] - 7s 15ms/step - loss: 0.6646 - ac
Epoch 5/20
469/469 [==============================] - 8s 17ms/step - loss: 0.6660 - ac
Epoch 6/20
469/469 [==============================] - 7s 15ms/step - loss: 0.6497 - ac
Epoch 7/20
469/469 [==============================] - 8s 17ms/step - loss: 0.6459 - ac
Epoch 8/20
469/469 [==============================] - 7s 16ms/step - loss: 0.6198 - ac
Epoch 9/20
469/469 [==============================] - 7s 16ms/step - loss: 0.6197 - ac
Epoch 10/20
469/469 [==============================] - 7s 16ms/step - loss: 0.6126 - ac
Epoch 11/20
469/469 [==============================] - 7s 16ms/step - loss: 0.6014 - ac
Epoch 12/20
469/469 [==============================] - 7s 16ms/step - loss: 0.6095 - ac
Epoch 13/20
469/469 [==============================] - 7s 15ms/step - loss: 0.6105 - ac
Epoch 14/20
469/469 [==============================] - 7s 15ms/step - loss: 0.6063 - ac
Epoch 15/20
469/469 [==============================] - 7s 16ms/step - loss: 0.5914 - ac
Epoch 16/20
469/469 [==============================] - 8s 17ms/step - loss: 0.5791 - ac
Epoch 17/20
469/469 [==============================] - 7s 15ms/step - loss: 0.5743 - ac
Epoch 18/20
469/469 [==============================] - 8s 17ms/step - loss: 0.5767 - ac
Epoch 19/20
```

50% Training Data

```
Test accuracy: 72.65
```

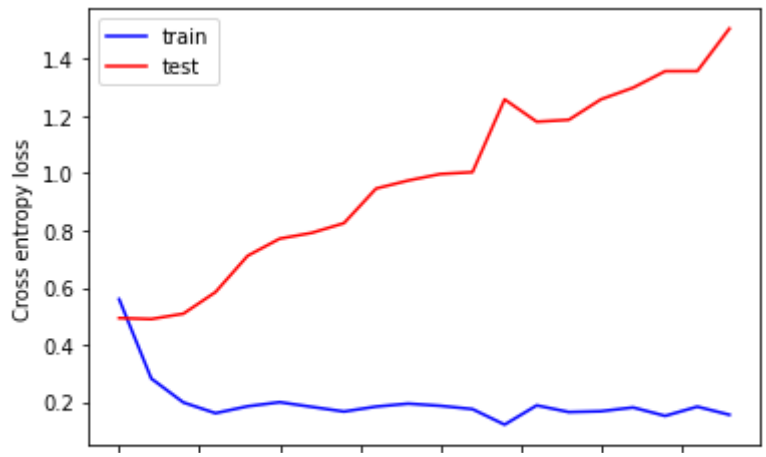
```
# ======= 50% Training Data ======


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
```

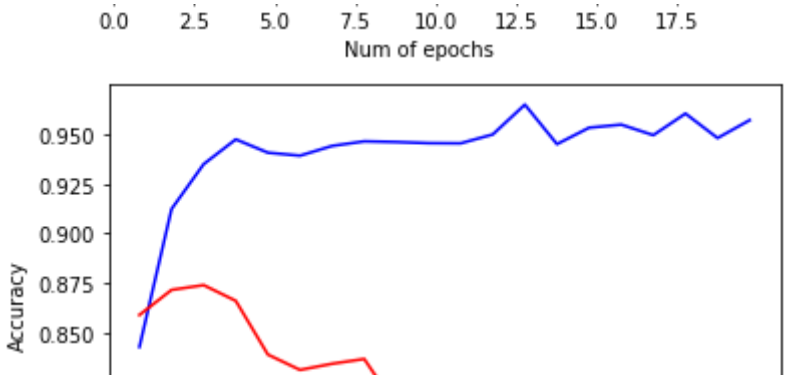
```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2)

X_train, X_test75, y_train, y_test75 = train_test_split(X_train, y_train, train_size=(
```

MODEL 5 : 50% Training Data | No Regularization

```
# Time how fast the model train

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.")

# Loss and Accuracy

loss, acc = model_1.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
313/313 [==============================] - 6s 18ms/step - loss: 0.5597 - ac
Epoch 2/20
313/313 [==============================] - 5s 14ms/step - loss: 0.2823 - ac
Epoch 3/20
313/313 [==============================] - 5s 14ms/step - loss: 0.1986 - ac
Epoch 4/20
313/313 [==============================] - 5s 15ms/step - loss: 0.1611 - ac
Epoch 5/20
313/313 [==============================] - 5s 15ms/step - loss: 0.1852 - ac
Epoch 6/20
313/313 [==============================] - 5s 15ms/step - loss: 0.1995 - ac
Epoch 7/20
313/313 [==============================] - 6s 18ms/step - loss: 0.1831 - ac
Epoch 8/20
313/313 [==============================] - 6s 20ms/step - loss: 0.1668 - ac
Epoch 9/20
313/313 [==============================] - 6s 18ms/step - loss: 0.1841 - ac
Epoch 10/20
313/313 [==============================] - 6s 19ms/step - loss: 0.1945 - ac
Epoch 11/20
313/313 [==============================] - 5s 16ms/step - loss: 0.1869 - ac
Epoch 12/20
313/313 [==============================] - 6s 18ms/step - loss: 0.1753 - ac
Epoch 13/20
313/313 [==============================] - 5s 17ms/step - loss: 0.1211 - ac
Epoch 14/20
313/313 [==============================] - 5s 16ms/step - loss: 0.1883 - ac
Epoch 15/20
313/313 [==============================] - 6s 18ms/step - loss: 0.1650 - ac
Epoch 16/20
313/313 [==============================] - 6s 18ms/step - loss: 0.1679 - ac
Epoch 17/20
313/313 [==============================] - 6s 19ms/step - loss: 0.1810 - ac
Epoch 18/20
313/313 [==============================] - 6s 19ms/step - loss: 0.1516 - ac
Epoch 19/20
313/313 [==============================] - 6s 19ms/step - loss: 0.1839 - ac
Epoch 20/20
313/313 [==============================] - 6s 18ms/step - loss: 0.1555 - ac
Total training time: 1.7974004904429117 minutes.
Test loss: 3.5814
Test accuracy: 64.06
```

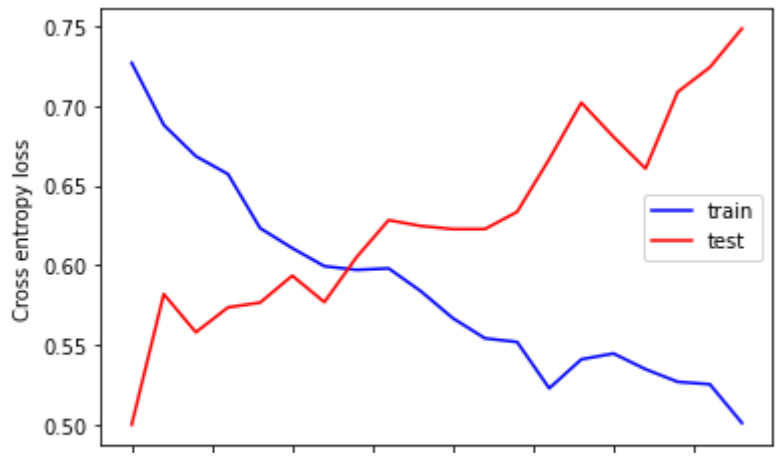


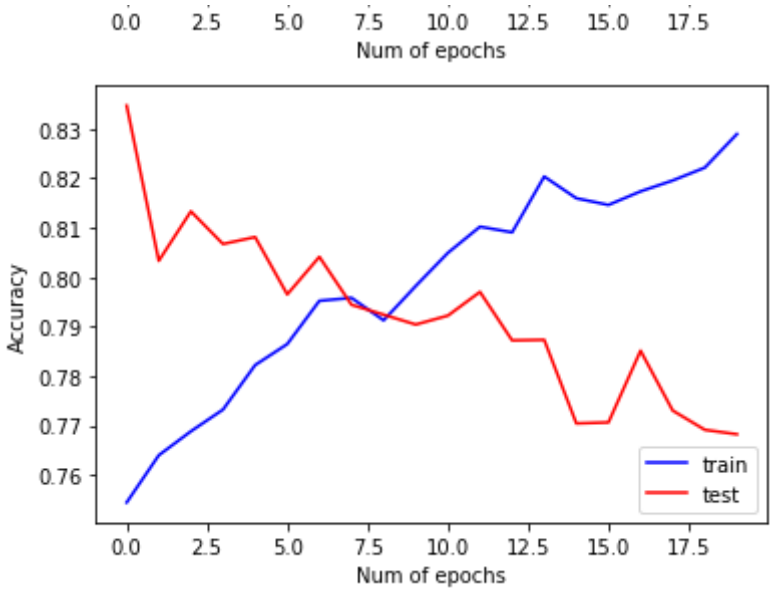


MODEL 6 : 50% Training Data | With Regularization

```
# Time how fast the model train

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.")

# 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()
```

```
Epoch 1/20
313/313 [==============================] - 6s 19ms/step - loss: 0.7269 - ac
Epoch 2/20
313/313 [==============================] - 6s 18ms/step - loss: 0.6879 - ac
Epoch 3/20
313/313 [==============================] - 5s 17ms/step - loss: 0.6683 - ac
Epoch 4/20
313/313 [==============================] - 6s 18ms/step - loss: 0.6571 - ac
Epoch 5/20
313/313 [==============================] - 6s 18ms/step - loss: 0.6231 - ac
Epoch 6/20
313/313 [==============================] - 5s 16ms/step - loss: 0.6106 - ac
Epoch 7/20
313/313 [==============================] - 5s 17ms/step - loss: 0.5992 - ac
Epoch 8/20
313/313 [==============================] - 6s 18ms/step - loss: 0.5969 - ac
Epoch 9/20
313/313 [==============================] - 5s 16ms/step - loss: 0.5979 - ac
Epoch 10/20
313/313 [==============================] - 5s 17ms/step - loss: 0.5836 - ac
Epoch 11/20
313/313 [==============================] - 6s 18ms/step - loss: 0.5666 - ac
Epoch 12/20
313/313 [==============================] - 6s 18ms/step - loss: 0.5539 - ac
Epoch 13/20
313/313 [==============================] - 5s 17ms/step - loss: 0.5517 - ac
Epoch 14/20
313/313 [==============================] - 6s 18ms/step - loss: 0.5226 - ac
Epoch 15/20
313/313 [==============================] - 6s 18ms/step - loss: 0.5408 - ac
Epoch 16/20
313/313 [==============================] - 5s 16ms/step - loss: 0.5444 - ac
Epoch 17/20
313/313 [==============================] - 5s 17ms/step - loss: 0.5345 - ac
Epoch 18/20
313/313 [==============================] - 5s 17ms/step - loss: 0.5266 - ac
Epoch 19/20
313/313 [==============================] - 5s 16ms/step - loss: 0.5251 - ac
Epoch 20/20
313/313 [==============================] - 6s 18ms/step - loss: 0.5008 - ac
Total training time: 2.3696436643600465 minutes.
Test loss: 1.0095
Test accuracy: 71.75
```





✓ 3m 24s completed at 11:50 AM