cnn-implementation

June 8, 2023

CNN implementation using CIFAR-10 dataset

[22]:
```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

[23]:
```
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.
↪load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0
```

[24]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
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(64, (3, 3), activation='relu'))
```

[25]:
```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

[26]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.
↪SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

[29]:
```
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

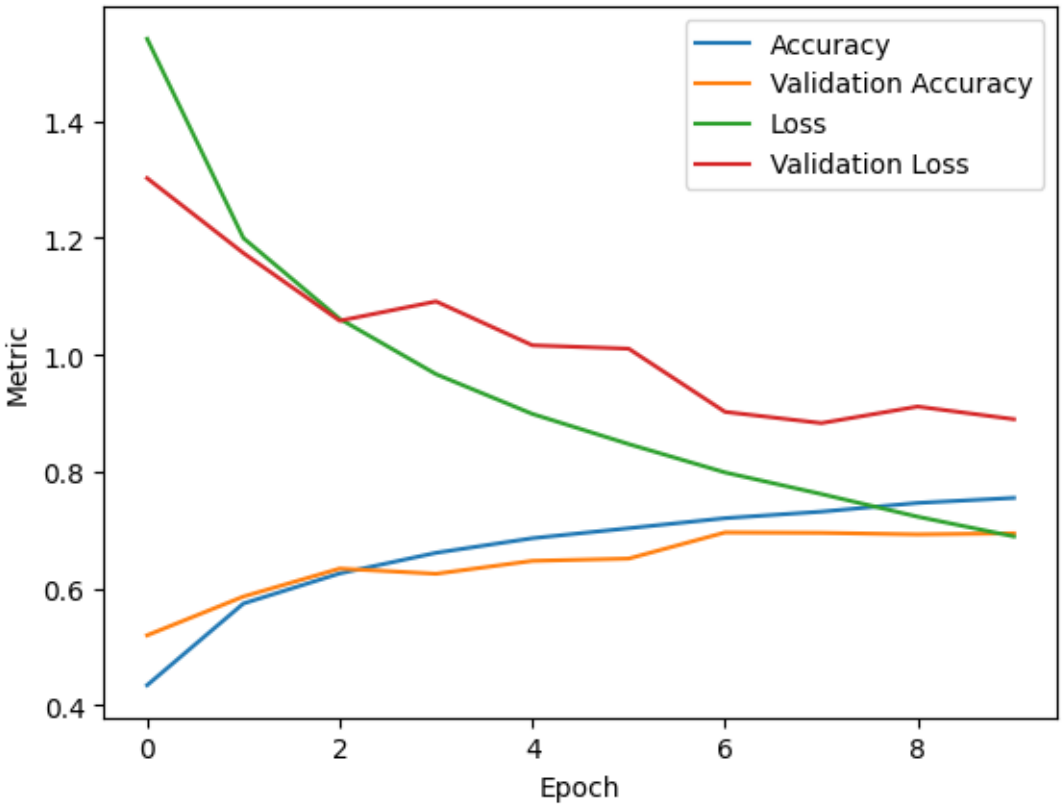
```
Epoch 1/10
1563/1563 [==============================] - 97s 61ms/step - loss: 1.5412 -
accuracy: 0.4339 - val_loss: 1.3025 - val_accuracy: 0.5194
Epoch 2/10
1563/1563 [==============================] - 84s 54ms/step - loss: 1.1998 -
accuracy: 0.5739 - val_loss: 1.1742 - val_accuracy: 0.5858
Epoch 3/10
1563/1563 [==============================] - 82s 53ms/step - loss: 1.0615 -
accuracy: 0.6254 - val_loss: 1.0582 - val_accuracy: 0.6337
```

```
Epoch 4/10
1563/1563 [==============================] - 84s 54ms/step - loss: 0.9667 -
accuracy: 0.6607 - val_loss: 1.0914 - val_accuracy: 0.6248
Epoch 5/10
1563/1563 [==============================] - 83s 53ms/step - loss: 0.8986 -
accuracy: 0.6857 - val_loss: 1.0162 - val_accuracy: 0.6469
Epoch 6/10
1563/1563 [==============================] - 81s 52ms/step - loss: 0.8470 -
accuracy: 0.7028 - val_loss: 1.0104 - val_accuracy: 0.6508
Epoch 7/10
1563/1563 [==============================] - 91s 58ms/step - loss: 0.7984 -
accuracy: 0.7200 - val_loss: 0.9022 - val_accuracy: 0.6956
Epoch 8/10
1563/1563 [==============================] - 85s 54ms/step - loss: 0.7614 -
accuracy: 0.7312 - val_loss: 0.8829 - val_accuracy: 0.6949
Epoch 9/10
1563/1563 [==============================] - 83s 53ms/step - loss: 0.7226 -
accuracy: 0.7461 - val_loss: 0.9112 - val_accuracy: 0.6923
Epoch 10/10
1563/1563 [==============================] - 81s 52ms/step - loss: 0.6888 -
accuracy: 0.7548 - val_loss: 0.8896 - val_accuracy: 0.6941
```

[30]:
```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print("Test accuracy:", test_acc)
```

```
313/313 - 5s - loss: 0.8896 - accuracy: 0.6941 - 5s/epoch - 17ms/step
Test accuracy: 0.694100022315979
```

[31]:
```
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Metric')
plt.legend()
plt.show()
```


Accuracy
Validation Accuracy
Loss
Validation Loss
Metric
Epoch
0.4
0.6
0.8
1.0
1.2
1.4
0
2
4
6
8