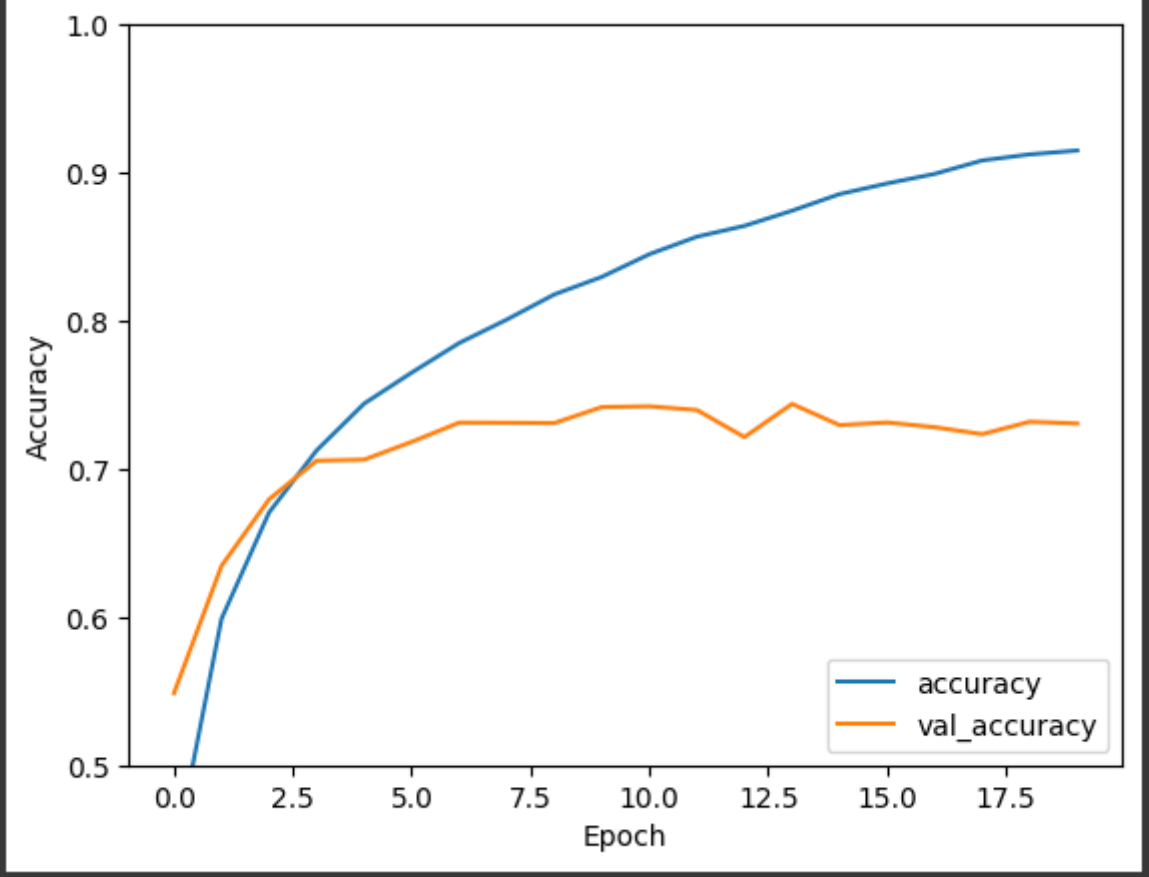
Model 2

```
# same baseline model but double up the layers

x = inputs = keras.Input(shape=(32, 32, 3))

x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)

x = layers.MaxPooling2D(pool_size=2)(x)

x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)

x = layers.MaxPooling2D(pool_size=2)(x)

x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)

x = layers.Flatten()(x)
x = layers.Dense(64, activation="relu")(x)
x = layers.Dense(10)(x)


outputs = x
model = keras.Model(inputs=inputs, outputs=outputs)
```

- Doubling up layers in the base model gets 73% accuracy

```
313/313 - 1s - loss: 1.1830 - accuracy: 0.7306 - 761ms/epoch - 2ms/step
```



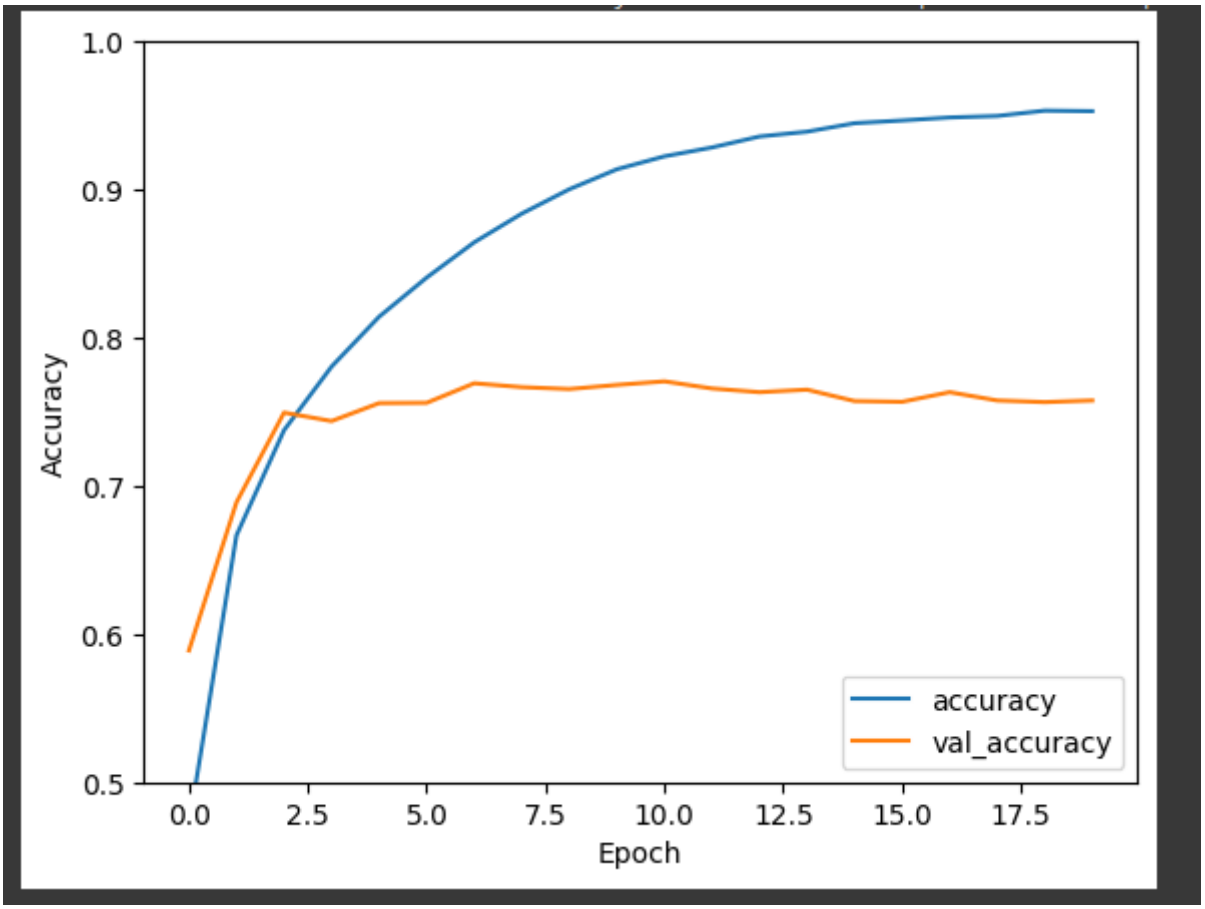
- Model overfits almost instantly
- Possibly add some dropout here

```
# remove this layer
#x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
```

- Remove the last Conv2D Layer
- **Accuracy 72%**

```
[11] x = inputs = keras.Input(shape=(32, 32, 3))
     #1
     x = layers.Conv2D(filters=32, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=32, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     #2
     x = layers.Conv2D(filters=64, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=64, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     #3
     x = layers.Conv2D(filters=128, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=128, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     #4
     x = layers.Flatten()(x)
     x = layers.Dense(128, activation="relu")(x)
     x = layers.Dense(10)(x)


     outputs = x
     model = keras.Model(inputs=inputs, outputs=outputs)
```

- **Almost 76% Accuracy
- Best performance yet



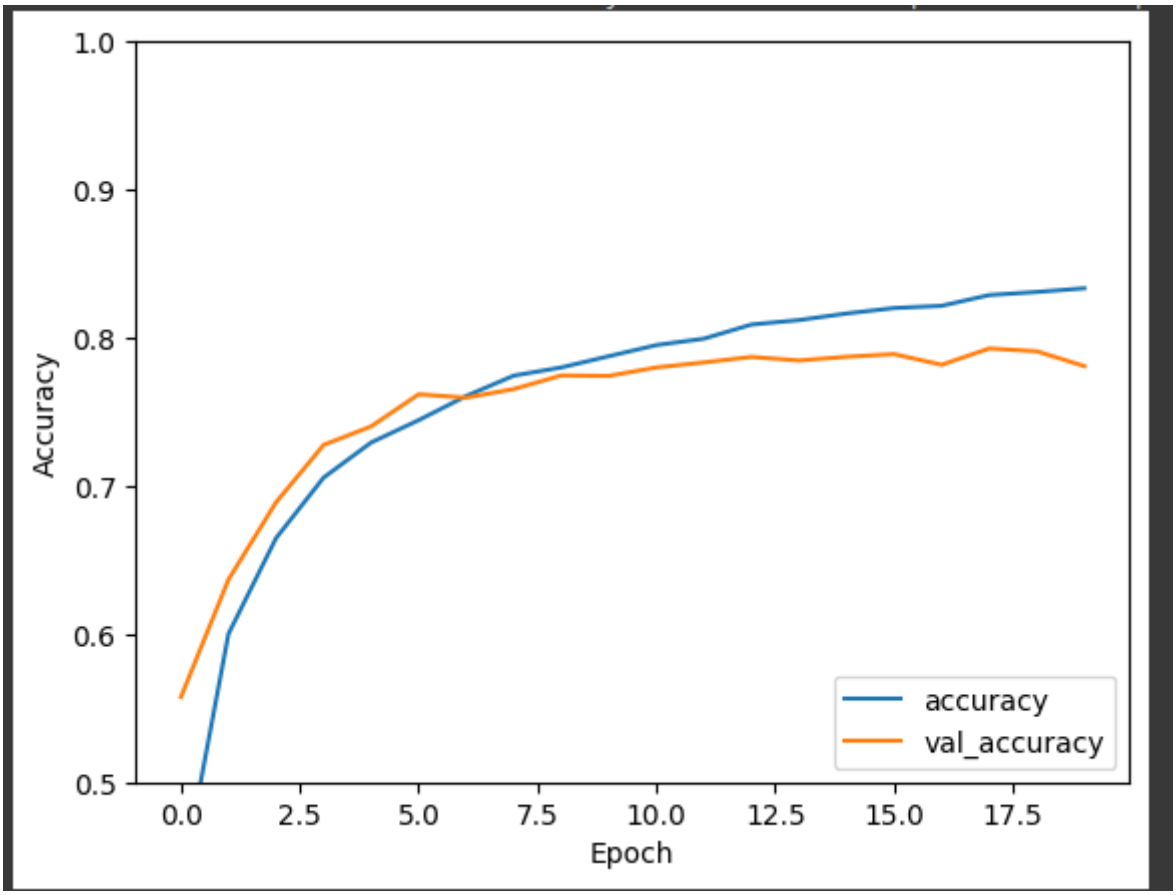
- However it still overfits

Can start to see, changing the layers is making much more of a difference

---

```
[18] x = inputs = keras.Input(shape=(32, 32, 3))
     #1
     x = layers.Conv2D(filters=32, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=32, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     x = layers.Dropout(0.25)(x)

     #2
     x = layers.Conv2D(filters=64, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=64, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     x = layers.Dropout(0.25)(x)

     #3
     x = layers.Conv2D(filters=128, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=128, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     x = layers.Dropout(0.25)(x)

     #4
     x = layers.Flatten()(x)
     x = layers.Dense(128, activation="relu")(x)
     x = layers.Dense(10)(x)


     outputs = x
     model = keras.Model(inputs=inputs, outputs=outputs)
```

- Adding in 3 Dropouts
- 1 after each pooling layer



- Improves accuracy by 2% to 78%
- Now takes more epochs before overfitting

```
[3]  x = inputs = keras.Input(shape=(32, 32, 3))
     #1
     x = layers.Conv2D(filters=32, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=32, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     x = layers.Dropout(0.1)(x)

     #2
     x = layers.Conv2D(filters=64, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=64, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     x = layers.Dropout(0.2)(x)

     #3
     x = layers.Conv2D(filters=128, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=128, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     x = layers.Dropout(0.4)(x)

     #4
     x = layers.Flatten()(x)
     x = layers.Dense(128, activation="relu")(x)
     x = layers.Dropout(0.5)(x)
     x = layers.Dense(10)(x)


     outputs = x
     model = keras.Model(inputs=inputs, outputs=outputs)
```

- Increase the dropout deeper in the network
- reduces a neurons reliance on each other
- Too much dropout here

```
[9]  x = inputs = keras.Input(shape=(32, 32, 3))
     #1
     x = layers.Conv2D(filters=32, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=32, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     x = layers.Dropout(0.2)(x)

     #2
     x = layers.Conv2D(filters=64, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=64, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     x = layers.Dropout(0.3)(x)

     #3
     x = layers.Conv2D(filters=128, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.Conv2D(filters=128, kernel_size=3, activation="relu", padding="same")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)
     x = layers.Dropout(0.4)(x)

     #4
     x = layers.Flatten()(x)
     x = layers.Dense(128, activation="relu")(x)
     x = layers.Dense(10)(x)


     outputs = x
     model = keras.Model(inputs=inputs, outputs=outputs)
```

- Slowly increase the dropout
- Model got to 80%

Adding augmentation on top of Dropout
Model underfits