Model 1

- Optimiser = Adam
- Epochs = 20

Model

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

     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.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)
```

- **Accuracy: 70%**
- Model is from Kaggle
- No Dropout / Augmentation

```
######
# with dropout after each pooling layer
######
# 3 dropout layers

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

#1
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(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")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Dropout(0.25)(x)

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

x = layers.Flatten()(x)
x = layers.Dense(64, activation="relu")(x)
x = layers.Dropout(0.25)(x)

x = layers.Dense(10)(x)


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

- Same Model but with 3 drop out layers
- One after each pooling layer
- And one at the end before the last dense layer
- 67% accuracy
- Too many drop out layers

```
[8] ######
    # dropout after each pooling layer
    ######
    # 2 dropout layers

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

    #1
    x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(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")(x)
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Dropout(0.25)(x)

    #3
    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)
```

- 2 dropout layers
- 1 after each pooling layer
- Removed the third dropout layer at the end
- **Accuracy: 74%**
- Removing the dropout at the end increases the accuracy

```
######
# dropout after each pooling layer
######
# 2 dropout layers
# Increased drop out to 0.5

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

#1
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Dropout(0.5)(x)

#2
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Dropout(0.5)(x)

#3
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)
```

- increase the 2 dropout layers to 0.5
- **Accuracy 71%**
- perhaps too much dropout

```
[12] ######
     # dropout at the end
     ######
     # 1 dropout layers

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

     #1
     x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)

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

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

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

     x = layers.Dropout(0.25)(x)
     x = layers.Dense(10)(x)


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

- 1 dropout layer
- at the end before the last dense layer
- **Accuracy 72.6%**

```
[16] ######
     # dropout at the end
     ######
     # 1 dropout layers
     # increase the drop out ot 0.5

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

     #1
     x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
     x = layers.MaxPooling2D(pool_size=2)(x)

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

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

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

     x = layers.Dropout(0.5)(x)
     x = layers.Dense(10)(x)


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

- 1 dropout layer
- Increased the Dropout to 0.5
- at the end before the last dense layer
- Almost no increase in accuracy
- **Accuracy 72.8%**

```
[32]
    data_augmentation = keras.Sequential(
        [
            layers.RandomFlip("horizontal"),
            layers.RandomRotation(0.1),
            layers.RandomZoom(0.2),
        ]
    )

    x = inputs = keras.Input(shape=(32, 32, 3))
    # 1
    # Augmentation Layer
    x = data_augmentation(inputs)
    # 2
    x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
    x = layers.MaxPooling2D(pool_size=2)(x)
    # 3
    x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
    x = layers.MaxPooling2D(pool_size=2)(x)
    # 4
    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)
```

- No Dropout layers
- 1 Augmentation layer
- Random flip, Random rotation, random zoom
- careful not to augment it too much
- **Accuracy: 68%**
- Possibly too much augmentation

```
[47]
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomZoom(0.2),
    ]
)

x = inputs = keras.Input(shape=(32, 32, 3))
# 1
# Augmentation Layer
x = data_augmentation(inputs)
# 2
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
# 3
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
# 4
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)
```

- Removed random rotation
- **Accuracy 74%**
- Nice improvement
- If the image is augmented too much the useful features from the images are gone

---

```
data_augmentation = keras.Sequential(
    [
        layers.RandomTranslation(height_factor=(-0.1, 0.1), width_factor=(-0.1, 0.1)),

    ]
)

x = inputs = keras.Input(shape=(32, 32, 3))
# 1
# Augmentation Layer
x = data_augmentation(inputs)
# 2
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
# 3
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
# 4
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)
```

- Only augmenting the width and height by 10%
- **Accuracy of 74%**

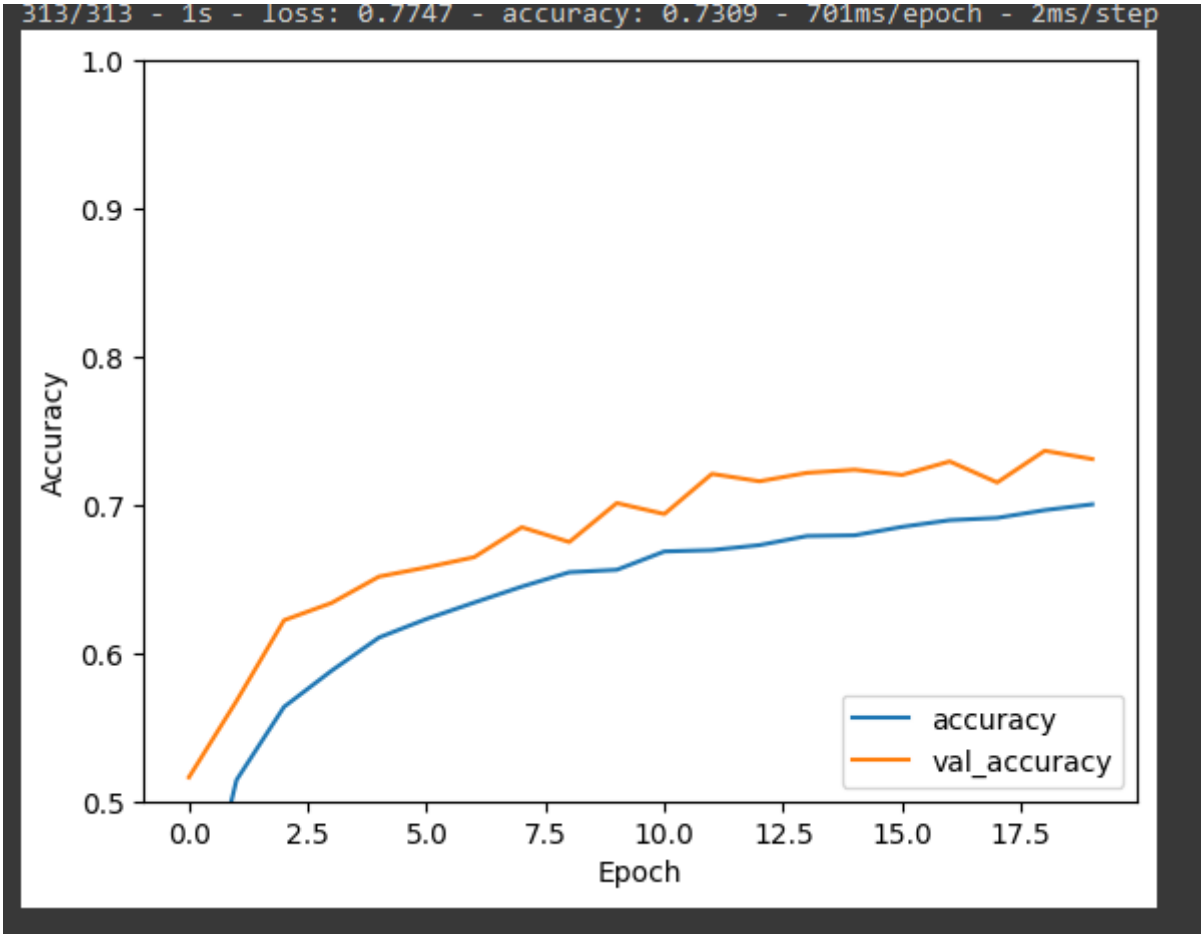
```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomZoom(0.2),
        layers.RandomTranslation(height_factor=(-0.1, 0.1), width_factor=(-0.1, 0.1)),

    ]
)
```

- Combine the two together
- **Accuracy 72%**
- Too much augmentation

```python
data_augmentation = keras.Sequential(
    [
        layers.RandomTranslation(height_factor=(-0.1, 0.1), width_factor=(-0.1, 0.1)),

    ]
)

x = inputs = keras.Input(shape=(32, 32, 3))
# 1
# Augmentation Layer
x = data_augmentation(inputs)
# 2
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Dropout(0.25)(x)

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

# 4
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)
```

- Combine the best dropout model and the best augmentation model
- Model starts to underfit
- Accuracy of 73%

```
313/313 - 1s - loss: 0.7747 - accuracy: 0.7309 - 701ms/epoch - 2ms/step
```



```python
data_augmentation = keras.Sequential(
    [
        layers.RandomTranslation(height_factor=(-0.1, 0.1), width_factor=(-0.1, 0.1)),

    ]
)

x = inputs = keras.Input(shape=(32, 32, 3))
# 1
# Augmentation Layer
x = data_augmentation(inputs)
# 2
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Dropout(0.25)(x)

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

# 4
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)
```

- Some Augmentation & 1 dropout layer
- Underfits
- **Accuracy: 67%**

Summary So Far

- 2 Dropout layers 1 after each pooling layer got 74%
- Image augmentation can get also get up to 74%
- Augmenting the width/height, or the zoom and image flip produce the best result
- Have to be careful not to dropout too much or augment too much

Takeaways

- Don't use too much dropout, careful where the dropout layers are. Too much dropout leads to underfitting
-