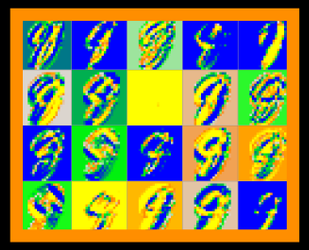
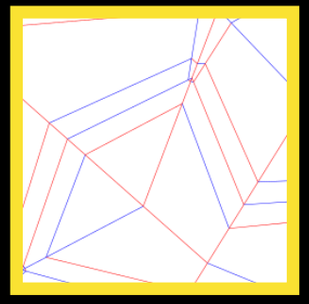
TensorFlow Tutorial, Part 1

Shruti Mishra and Jordan Hoffmann



Background

Setup for the aspects we will vary in the coming three demos



Demo 1

Here, we try to predict how many times a sheet has been folded

Classification problem • Geometry • Modify example structure

Demo 2

We solve a PDE and try to predict something about the future

Regression problem • Side Stream • Batch Generator

Demo 3

Given a set of grayscale images, can we recolor them?

Image reconstruction problem • Fully convolutional

IMPORT PACKAGES

- TensorFlow
- Numpy
- Pandas
- Seaborn

DEFINE MODEL

- Different layers
- Sizes, etc

COMPILE MODEL

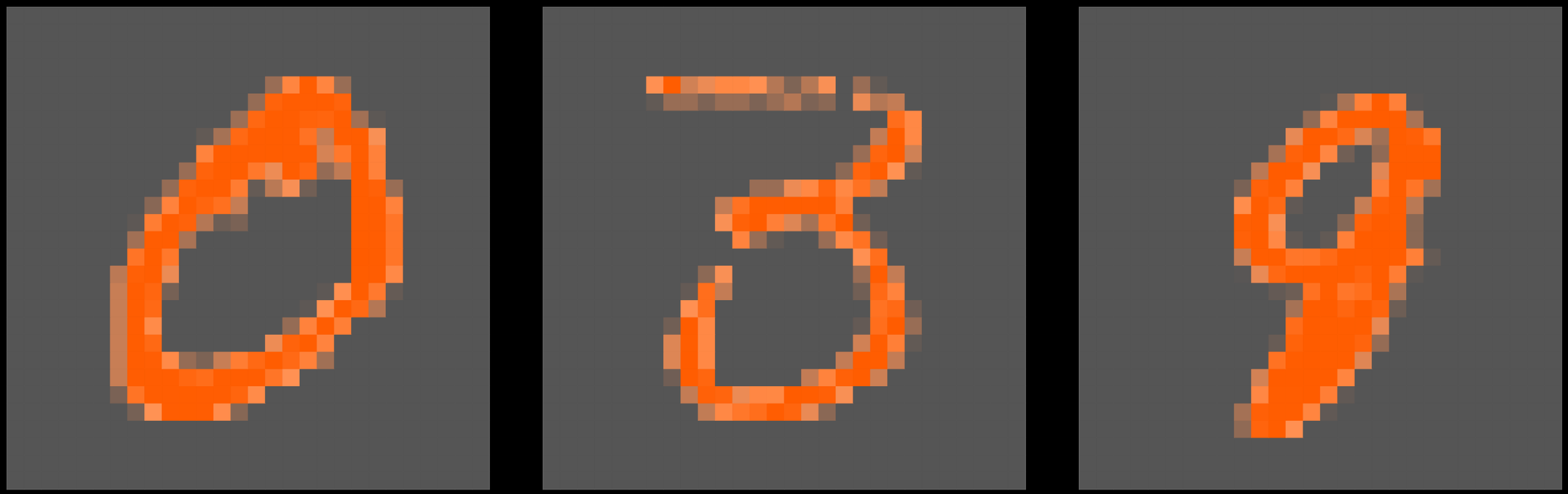
- Optimizer
- Loss
- Validation

FIT THE MODEL

- How long to train
- When to stop

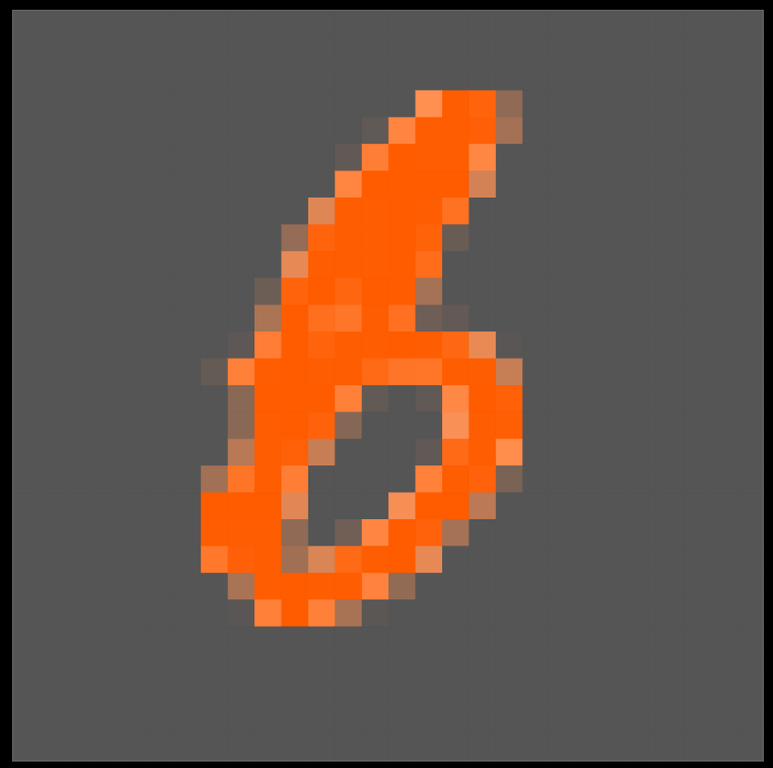
```
import tensorflow as tf

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(64, 64, 1)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])


model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=15)
```

```
import tensorflow as tf

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(64, 64, 1)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1, activation=tf.nn.relu)
])


model.compile(optimizer='sgd',
              loss='mean_squared_error')

model.fit(x_train, y_train, epochs=15, validation_set =(x_test,y_test))
```

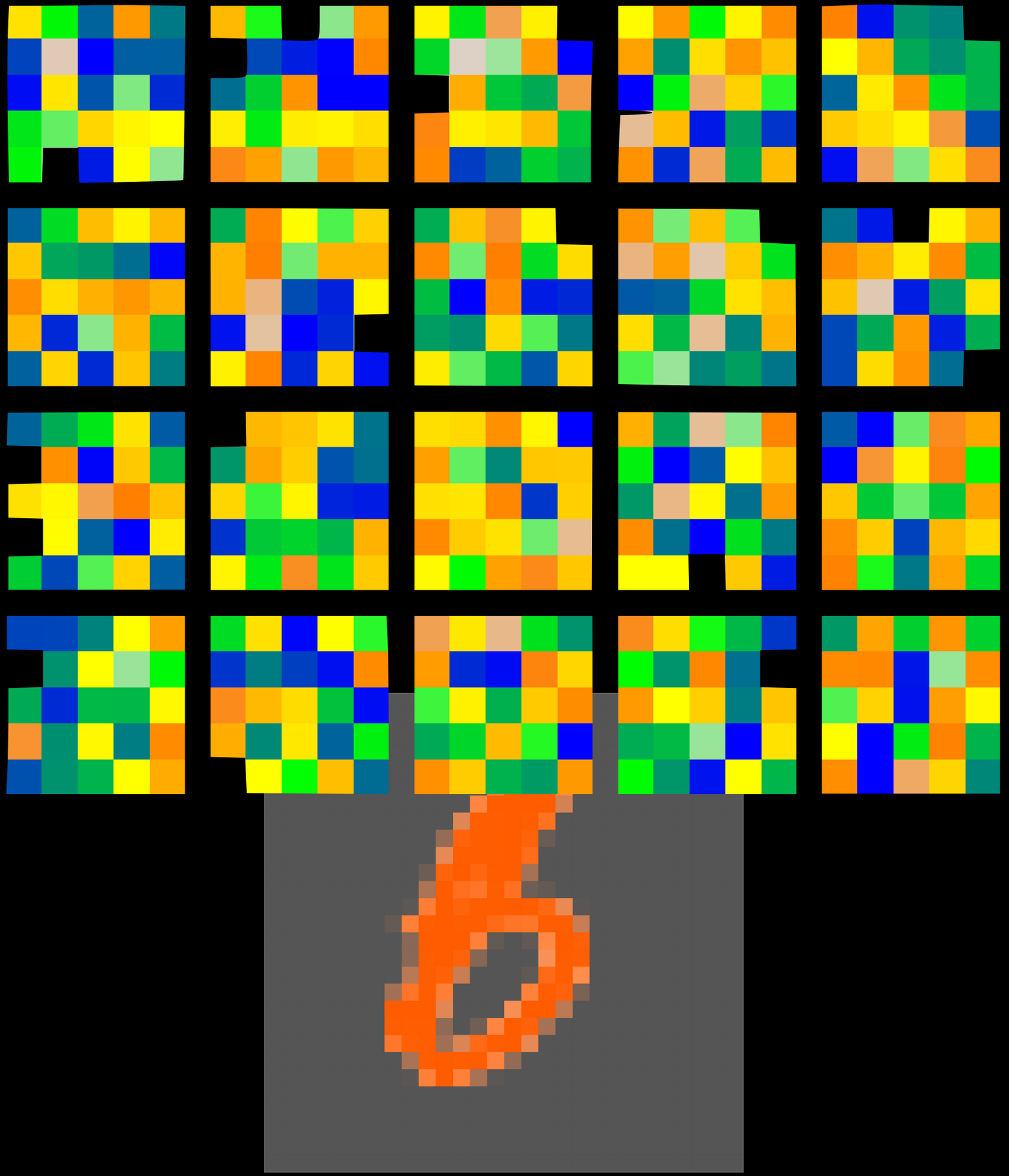


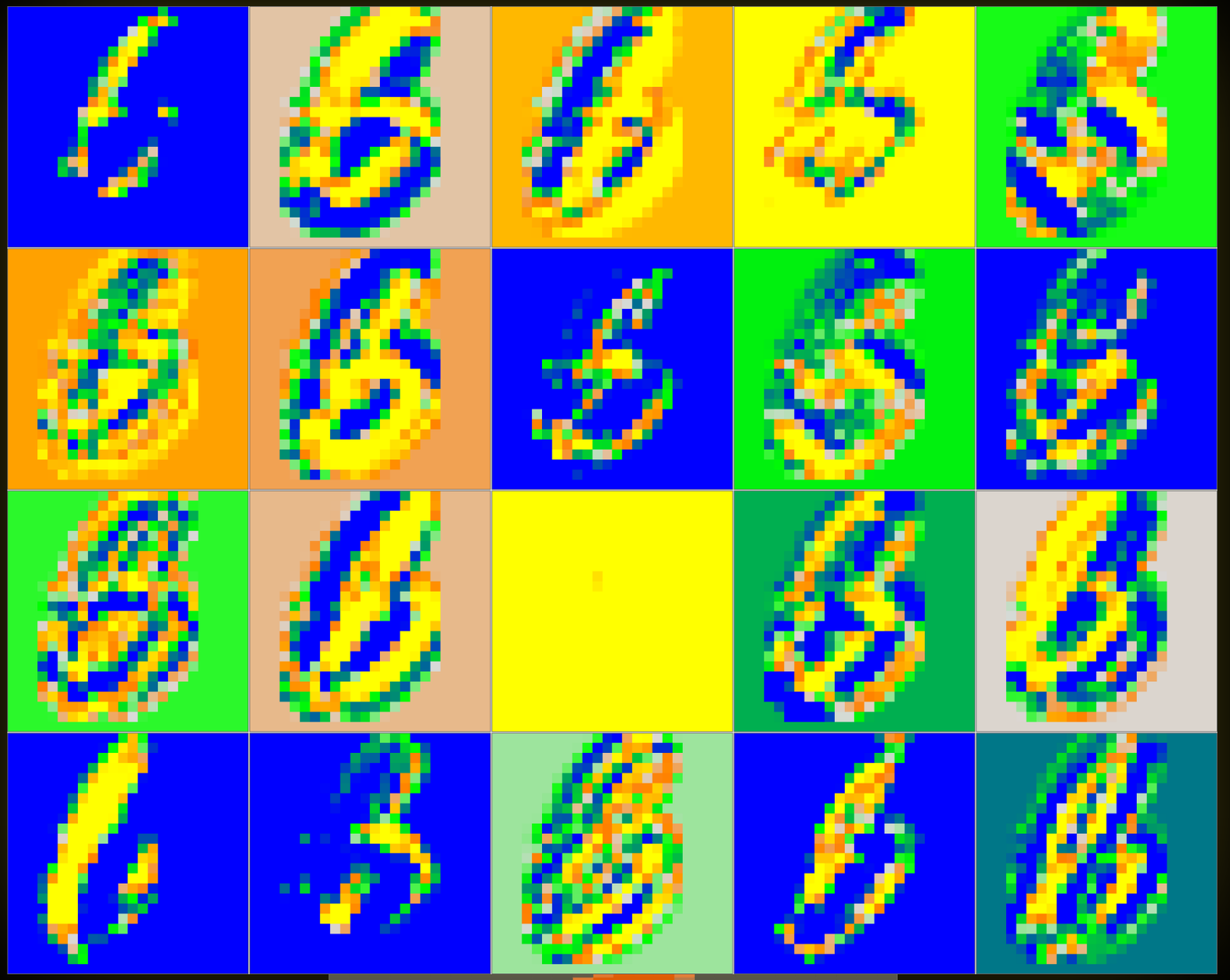
Softmax- logistic function

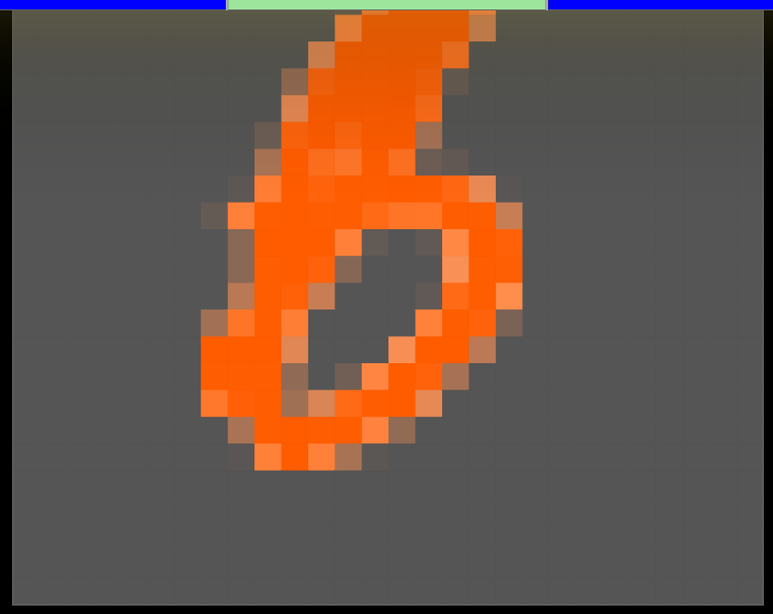
$$\sigma(\vec{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$



```
import tensorflow as tf

model = keras.Sequential([
    keras.layers.Conv2D(size=(3,3), filters=16, input_shape=(64, 64, 1)),
    keras.layers.MaxPool2D(pool_size=(2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1, activation=tf.nn.relu)
])


model.compile(optimizer='sgd',
              loss='mean_squared_error')

model.fit(x_train, y_train, epochs=15, validation_set =(x_test,y_test))
```



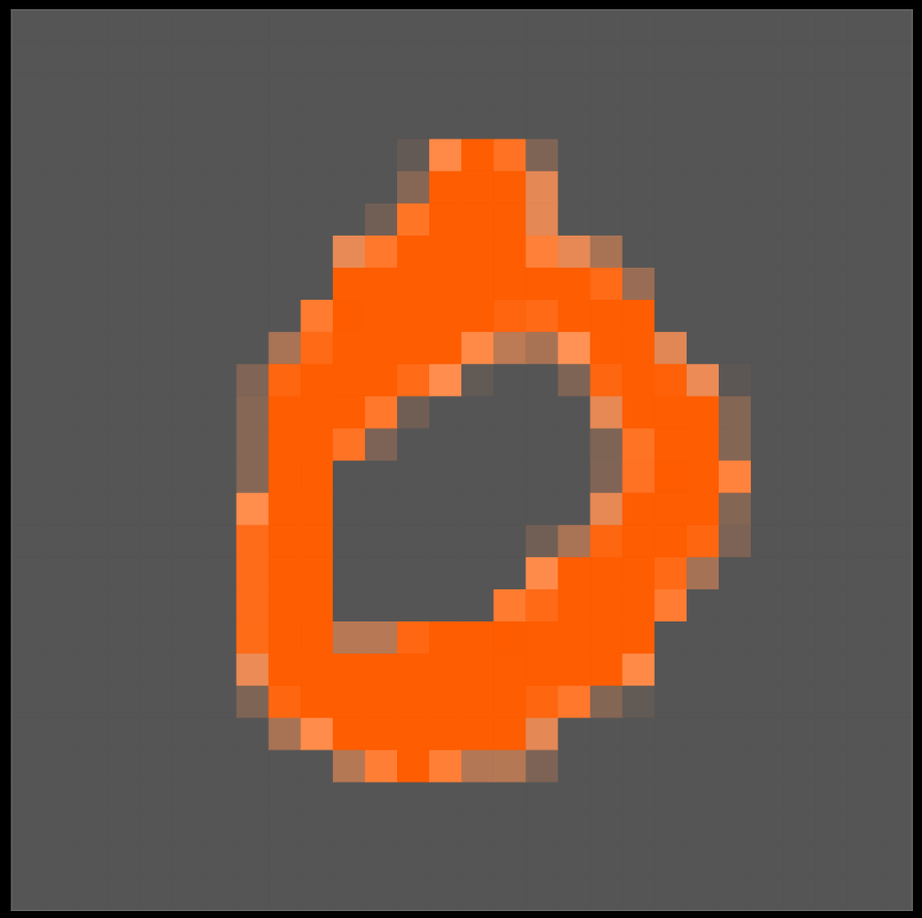


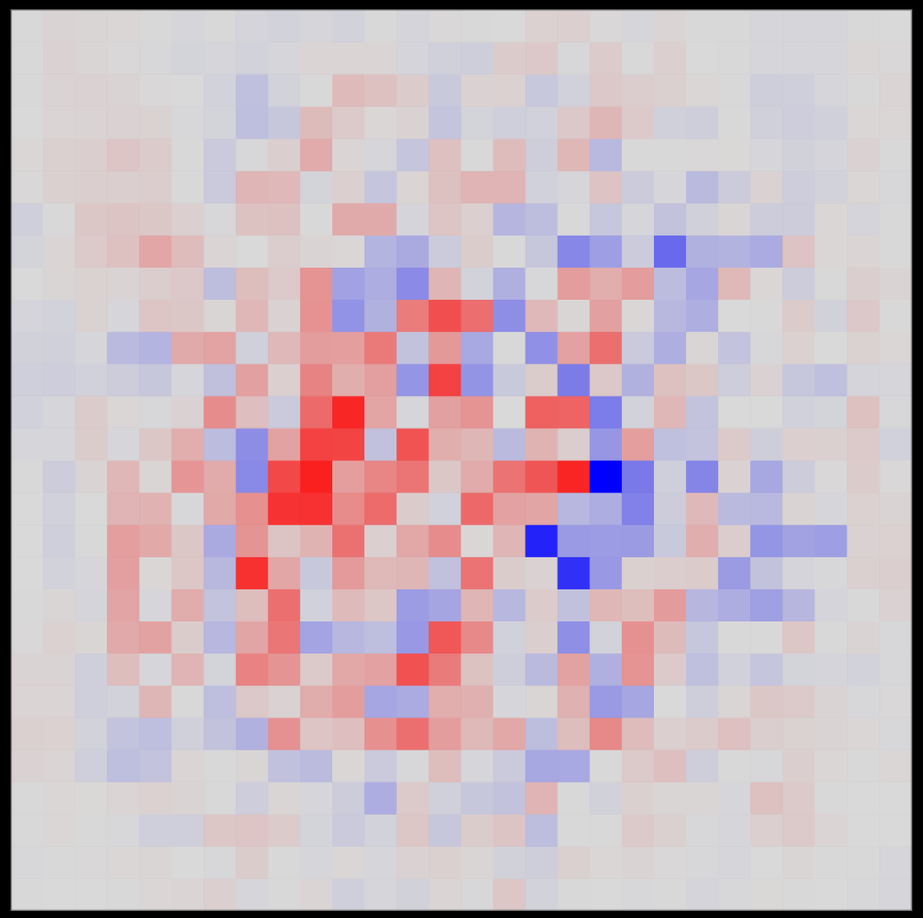


MaxPool2D

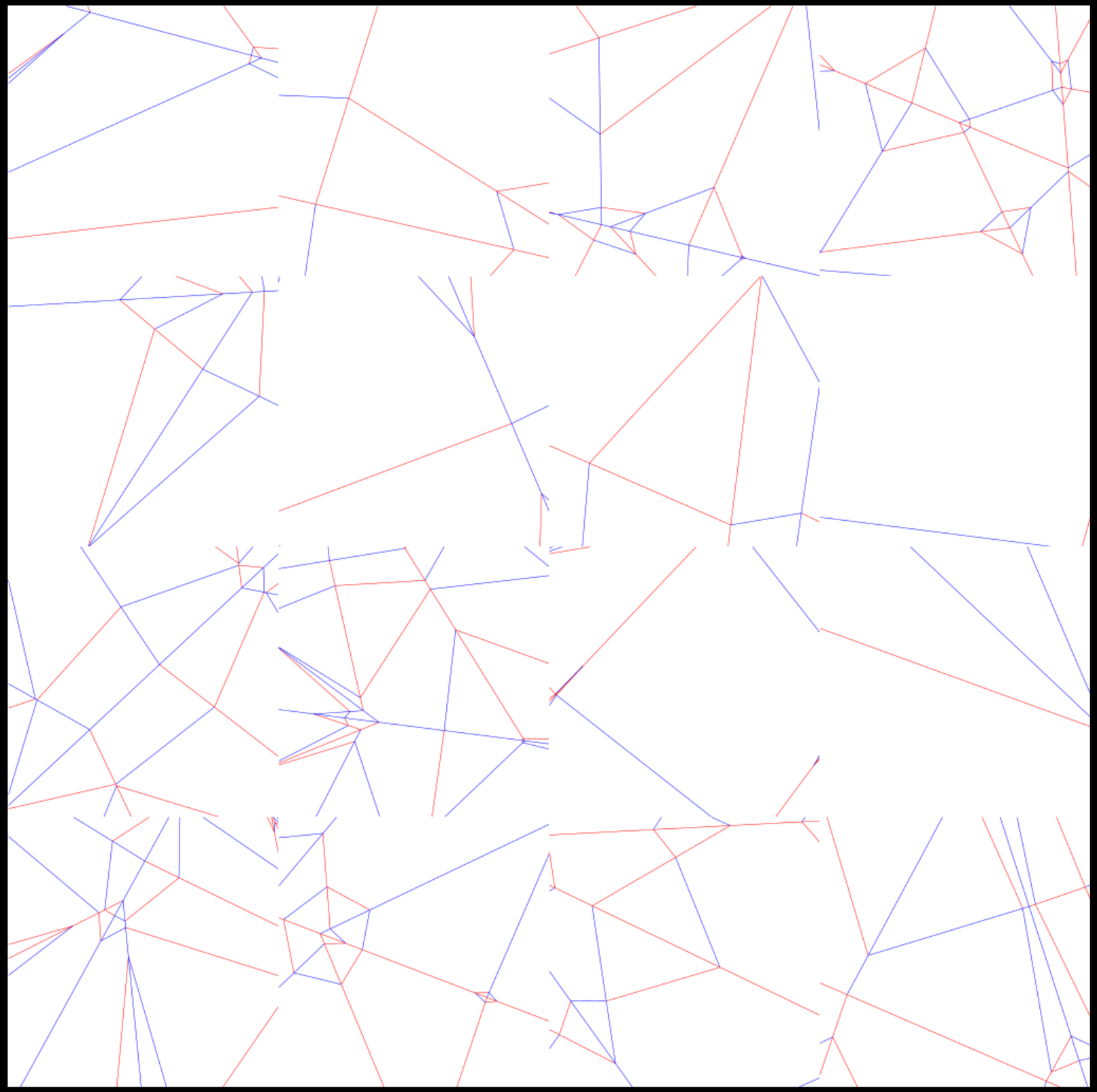
8	9	34	4
3	10	6	5
5	1	8	6
0	4	25	2

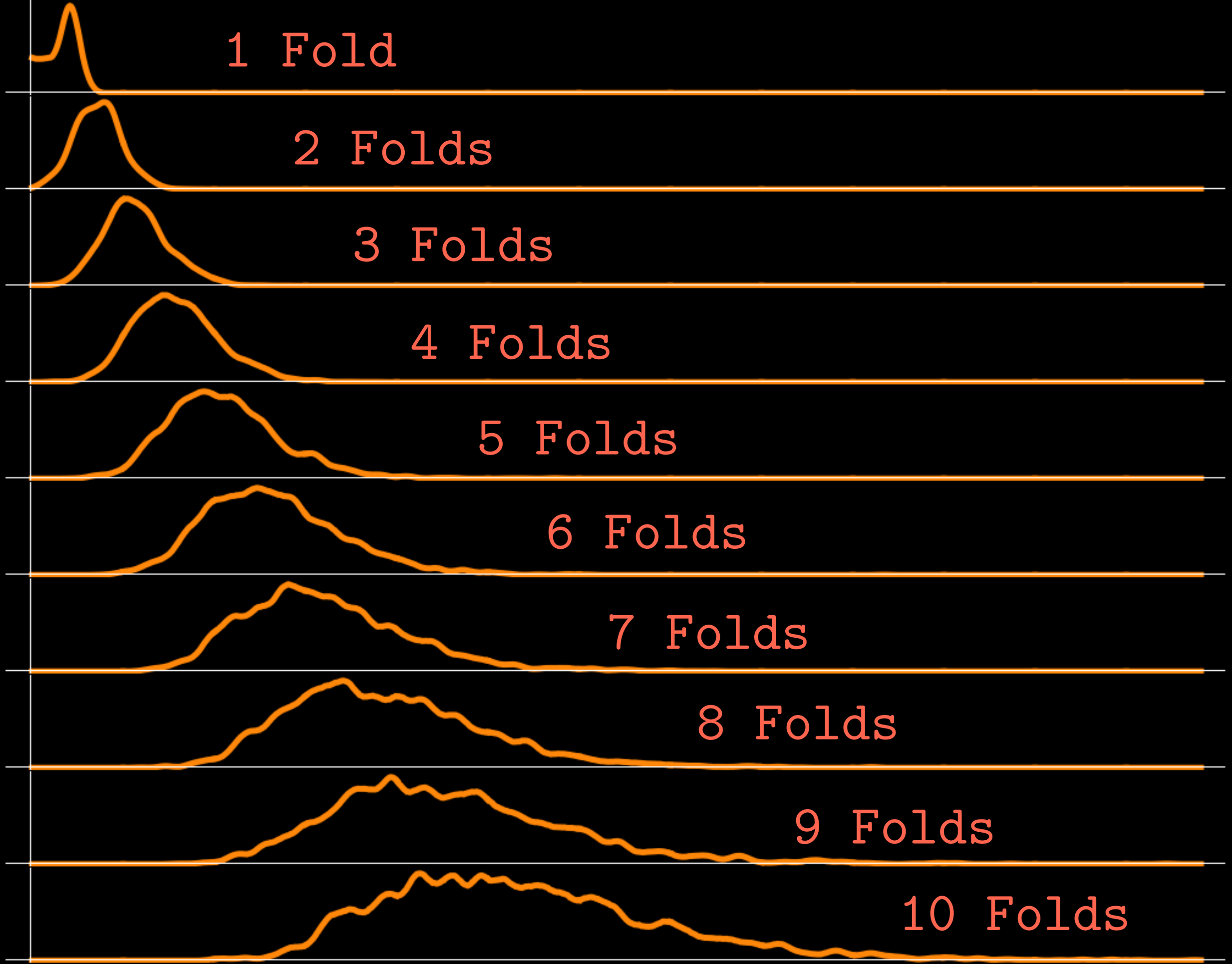
10	34
5	25

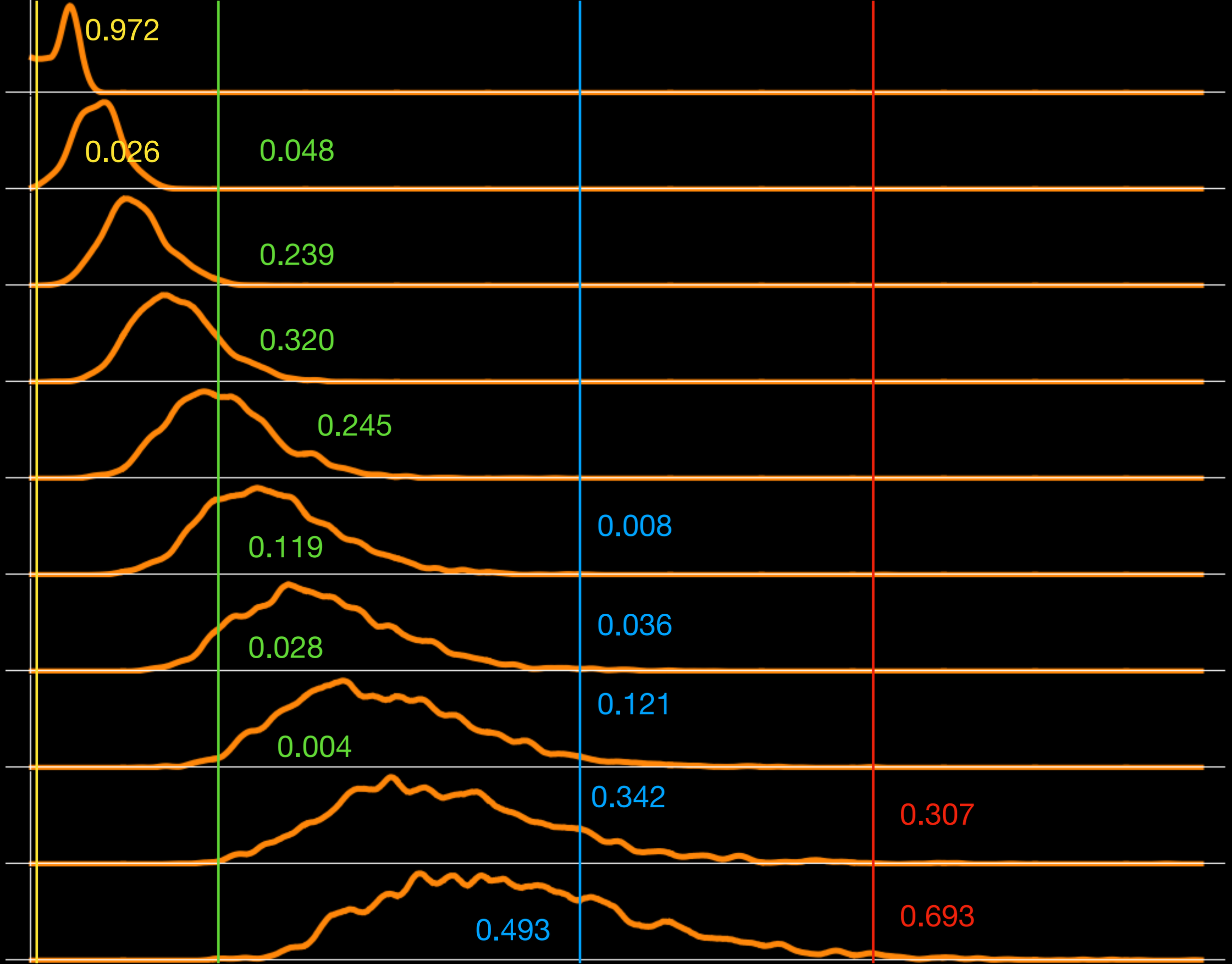


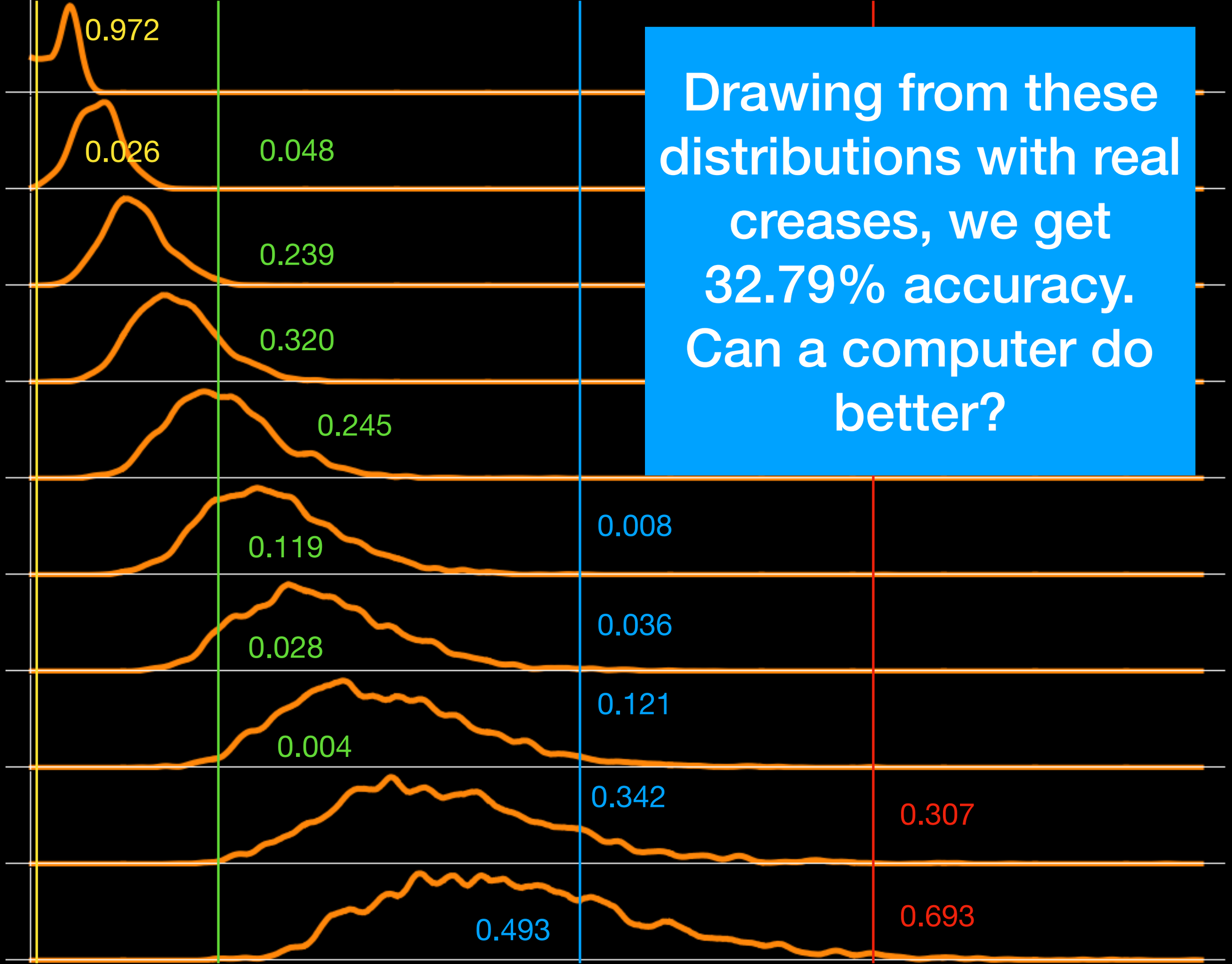


Mileage distributions of flat folded sheets




1 Fold
2 Folds
3 Folds
4 Folds
5 Folds
6 Folds
7 Folds
8 Folds
9 Folds
10 Folds


0.972
0.026
0.048
0.239
0.320
0.245
0.119
0.008
0.028
0.036
0.004
0.121
0.342
0.307
0.493
0.693



Drawing from these distributions with real creases, we get 32.79% accuracy. Can a computer do better?