To Hot-Dog Or Not Hot-Dog

By JinZhao Su and Theodore Kim

Overview

Let us ask ourselves this physiological question. What is a Hot-Dog? Is it just a stick of meat between two buns?

Throughout time and history, many have debated what exactly makes a Hot-Dog a Hot-Dog. Many creative minds have come close to an abstract definition of what a Hot-Dog is. Many philosophers have debated that a Hot-Dog is so much more than that; It's a message.

To settle the debate on whether or not a Hot-Dog is actually a Hot-Dog or not, scholar JinZhao Su and Theodore Kim created a Neural network to mimic the brain in order to understand which features make a Hot-Dog, truely a Hot-Dog.

Understanding the problems

01 What features exactly makes a Hot-Dog, an actually Hot-Dog?

02 How exactly do you abstractly separate the informations from a image to decimal digits?

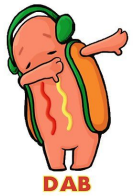
03 Philosophically what exactly does a Hot-Dog mean?

Persona 01

Extracting data:

The photos in category one are what we can agree upon as a 100% Hot-Dog. The photos in category two are what we either consider 100%, not Hot-Dog or a questionable/fake Hot-Dog. The two categories were then loaded into a list:


```
import os

hot_dog_image_dir = 'train/hot_dog/'
hot_dog_paths = [''.join(hot_dog_image_dir+filename) for filename in
                                os.listdir(hot_dog_image_dir)]

not_hot_dog_image_dir = 'train/not_hot_dog/'
not_hot_dog_paths = [''.join(not_hot_dog_image_dir+filename) for filename in
                                os.listdir(not_hot_dog_image_dir)]

image_paths_train = hot_dog_paths + not_hot_dog_paths
y_dog=np.ones((len(hot_dog_paths),1),dtype=int)
y_not=np.zeros((len(not_hot_dog_paths),1),dtype=int)
y_train=np.concatenate((y_dog, y_not), axis=0)
# print(image_paths_train)
# print(y_train)

#############################################################################################
#############
hot_dog_image_dir_test = 'test/hot_dog/'
hot_dog_paths_test = [''.join(hot_dog_image_dir_test+filename) for filename in
                                os.listdir(hot_dog_image_dir_test)]

not_hot_dog_image_dir_test = 'test/not_hot_dog/'
not_hot_dog_paths_test = [''.join(not_hot_dog_image_dir_test+filename) for filename in
                                os.listdir(not_hot_dog_image_dir_test)]

image_paths_test = hot_dog_paths_test + not_hot_dog_paths_test
y_dog_test=np.ones((len(hot_dog_paths_test),1),dtype=int)
y_dog_not=np.zeros((len(not_hot_dog_paths_test),1),dtype=int)
y_test=np.concatenate((y_dog_test, y_dog_not), axis=0)
# print(image_paths_test)
# print(y_test)
# print(y_test.shape)
```



Persona 01

Extracting data:


```
def loadpath(image_paths,xsize=16,ysize=16):
    X=[]
    for location in image_paths:
        img=load_img(location,target_size=(xsize,ysize))
        sx=img_to_array(img) #(512, 382, 3)
#         print(x.shape)
#         sx=sx.reshape((1,)+sx.shape) #(1, 512, 382, 3)
#         print(sx.shape)
#         print(sx)
        X.append(sx)
        input_shape=sx.shape
    return input_shape,np.array(X)
```

Persona 01

Extracting data:


```
from sklearn.model_selection import train_test_split

input_shape,x=loadpath(image_paths_train+image_paths_test,128,128)
y=np.concatenate((y_train, y_test), axis=0)
# from keras.utils import to_categorical
# y_train = to_categorical(y_train)
# y_test = to_categorical(y_test)

# y_train=y_train.transpose()
# y_test=y_test.transpose()

# y_train=np.matrix(y_train)
# y_test=np.matrix(y_test)

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42,shuffle=True)


print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Persona 01

Measurement of Accuracy

The accuracy measurement that was used was binary_accuracy instead of the normal categlory_accuracy. Here's the difference:

Binary_accuracy:

K.mean(K.equal(y_true, K.round(y_pred)))

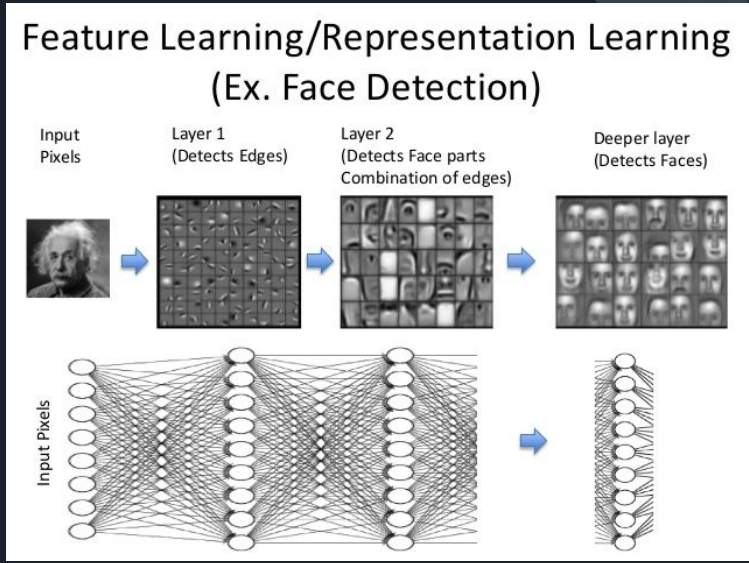
Categlory_accuracy:

K.mean(K.equal(K.argmax(y_true, axis=-1), K.argmax(y_pred, axis=-1)))

Persona 01

Understanding Relu; Convolutional2D

If the data was linear, then the formula would be y=w1*w2*w3*x. But because of the principle of Relu, you will need the equation to be reshaped to be around y=w3*max(0,w2*max(0, max(w1*x))). The advantage of using multiple layers is that it learns different representation at each layer. For example, a network that detects a human in an image, will learn edges in the first layer, shapes in the next, body parts in the next and finally humans in the last. This is precisely Model 2 was based upon. The ideology that by using two Convolutional 2D networks each time, you can gain maybe the left curve of the tip of the Hot-Dog and then zoomed out to see the tip of the Hot-Dog.



Persona 01

Models

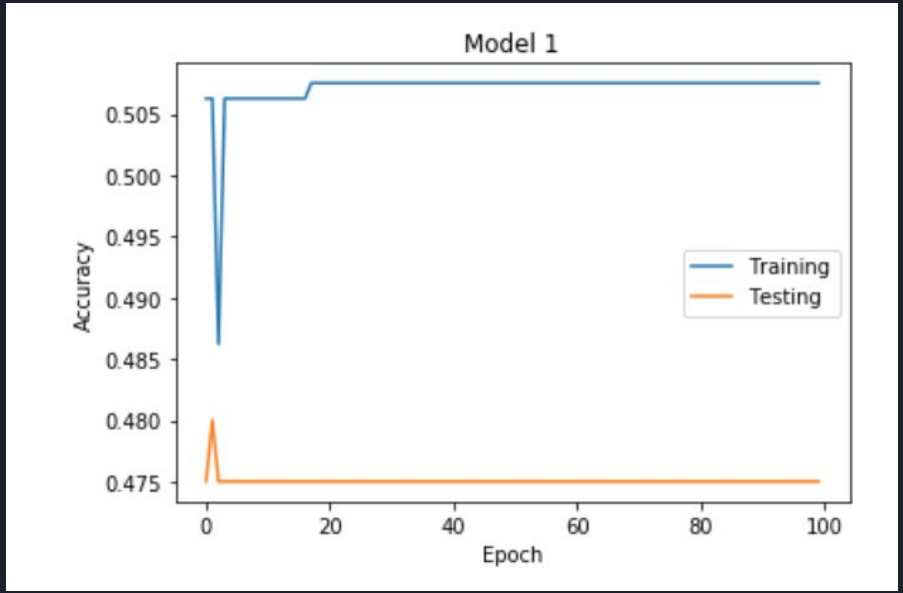

```
M1 = Sequential()

M1.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))

M1.add(Conv2D(64, (3, 3), activation='relu'))

M1.add(MaxPooling2D(pool_size=(3, 3)))

M1.add(Flatten())

M1.add(Dense(128, activation='relu'))

M1.add(Dense(1, activation='sigmoid'))

M1.summary()
```

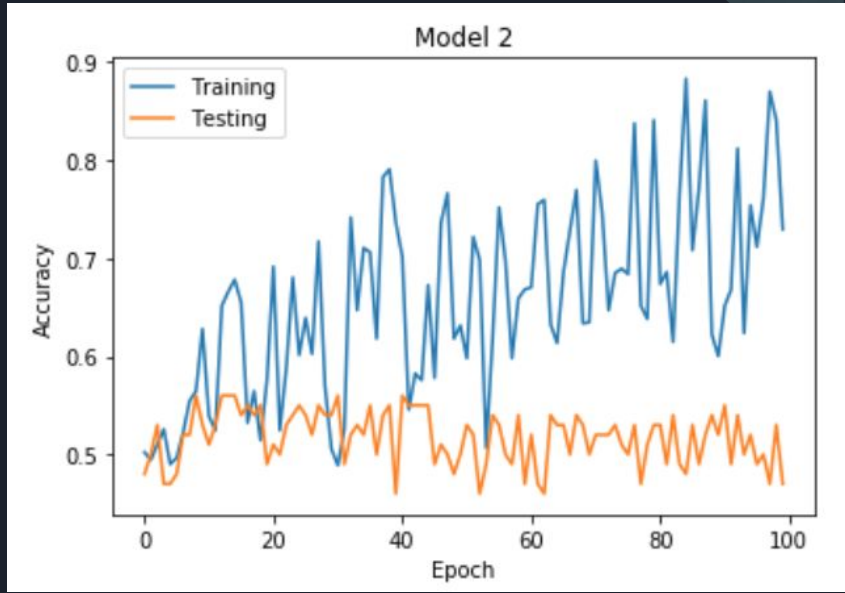
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 32)      896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 124, 124, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 41, 41, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 107584)            0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               13770880
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 13,790,401
Trainable params: 13,790,401
Non-trainable params: 0
_________________________________________________________________
```

```
M2 = Sequential()

M2.add(Conv2D(32, kernel_size=(3, 3),strides=(1,1),activation='relu',input_shap
M2.add(Conv2D(32, (3, 3), activation='relu'))

M2.add(MaxPooling2D(pool_size=(3, 3),strides=(2,2)))

M2.add(Conv2D(64, (3, 3), activation='relu'))
M2.add(Conv2D(64, (3, 3), activation='relu'))

M2.add(MaxPooling2D(pool_size=(4, 4)))

M2.add(Conv2D(128, (4, 4), activation='relu'))
M2.add(Conv2D(128, (4, 4), activation='relu'))

M2.add(MaxPooling2D(pool_size=(5, 5)))

M2.add(Flatten())

M2.add(Dense(2100,activation='relu'))

M2.add(Dense(1420,activation='relu'))

M2.add(Dense(128, activation='relu'))

M2.add(Dense(1, activation='sigmoid'))

M2.summary()
```

Persona 01

Models





Persona 01

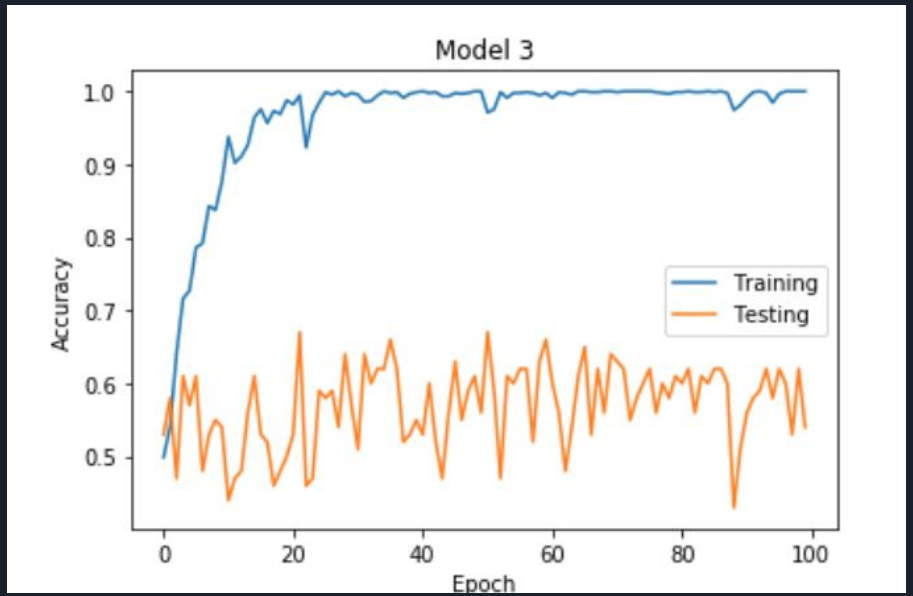
Models

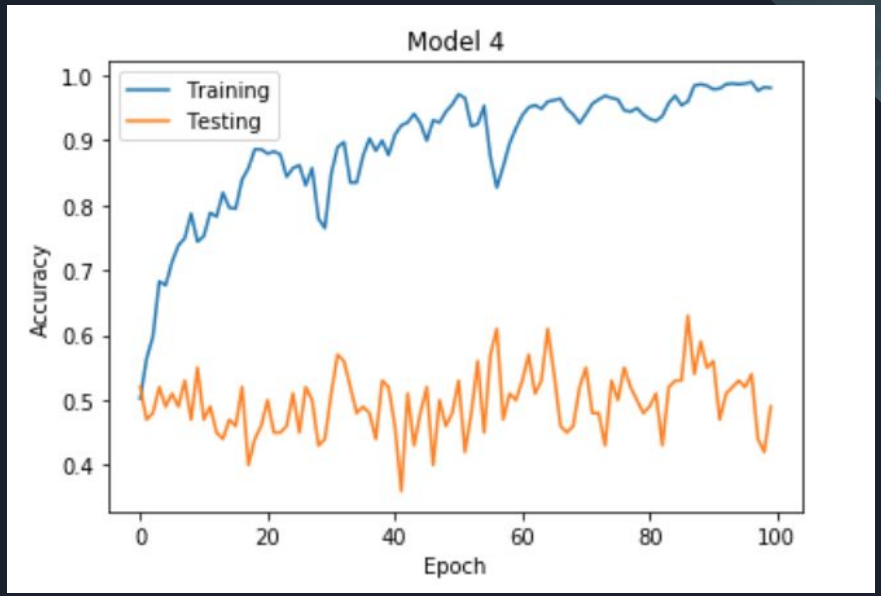

```
M3 = Sequential()

M3.add(Conv2D(32, kernel_size=(3, 3),strides=(1,1),activation='relu',input_shape=input_shape))
M3.add(Conv2D(64, (3, 3), activation='relu'))

M3.add(BatchNormalization())

M3.add(Dropout(.2))

M3.add(MaxPooling2D(pool_size=(3, 3),strides=(2,2)))

M3.add(Conv2D(64, (3, 3), activation='relu'))
M3.add(Conv2D(128, (3, 3), activation='relu'))

M3.add(Dropout(.2))

M3.add(BatchNormalization())

M3.add(MaxPooling2D(pool_size=(4, 4)))

M3.add(Conv2D(128, (4, 4), activation='relu'))
M3.add(Conv2D(128, (4, 4), activation='relu'))

M3.add(BatchNormalization())

M3.add(Dropout(.2))

M3.add(MaxPooling2D(pool_size=(4, 4)))

M3.add(Flatten())

M3.add(Dense(2100,activation='relu'))

M3.add(Dense(720,activation='relu'))

M3.add(Dense(1, activation='sigmoid'))

M3.summary()
```

```
M4 = Sequential()

M4.add(Conv2D(64, kernel_size=(3, 3),strides=(1,1),activation='relu',input_shape=input_shape))
M4.add(Conv2D(128, (2, 2), activation='relu'))

M4.add(MaxPooling2D(pool_size=(3, 3),strides=(2,2)))

M4.add(BatchNormalization())

M4.add(Dropout(.420))

M4.add(Conv2D(64, (1, 1), activation='relu'))
M4.add(Conv2D(64, (2, 2), activation='relu'))

M4.add(MaxPooling2D(pool_size=(4, 4)))

M4.add(Dropout(.420))

M4.add(BatchNormalization())

M4.add(Conv2D(64, (3, 3), activation='relu'))
M4.add(Conv2D(128, (2, 2), activation='relu'))

M4.add(MaxPooling2D(pool_size=(3, 3)))

M4.add(BatchNormalization())

M4.add(Dropout(.420))

M4.add(Flatten())

M4.add(Dense(1600,activation='relu'))

M4.add(BatchNormalization())

M4.add(Dense(720,activation='relu'))

M4.add(BatchNormalization())

M4.add(Dense(1, activation='sigmoid'))

M4.summary()
```

Persona 01

Models





Persona 01

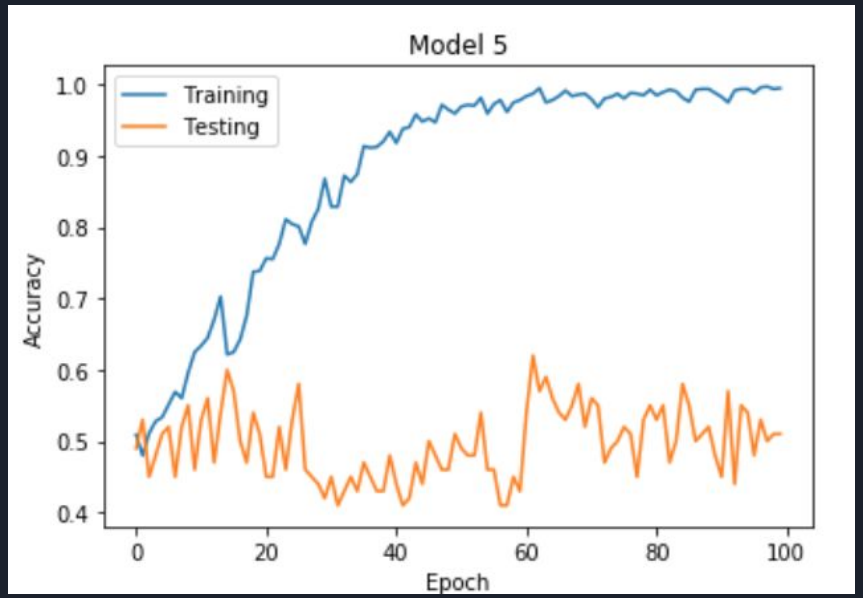
Models

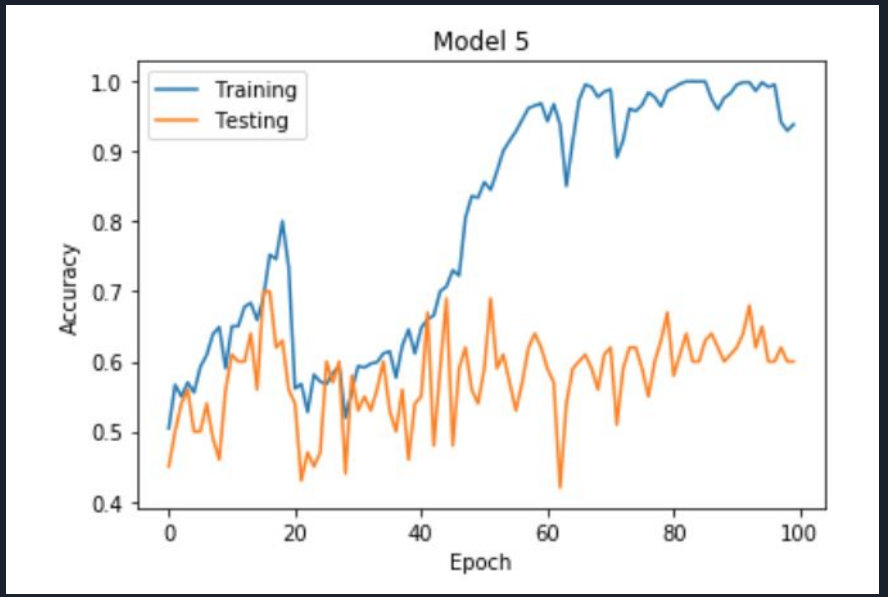
```
M5 = Sequential()

M5.add(Conv2D(32, kernel_size=(2, 2),strides=(1,1),activation='relu',input_shape=input_shape))

M5.add(BatchNormalization())

M5.add(Conv2D(32, (2, 2),strides=(2, 2), activation='relu'))

M5.add(BatchNormalization())

M5.add(MaxPooling2D(pool_size=(3, 3),strides=(2,2)))

M5.add(Dropout(.2))

M5.add(Conv2D(64, (3, 3), activation='relu'))

M5.add(BatchNormalization())

M5.add(Conv2D(64, (3, 3), activation='relu'))

M5.add(BatchNormalization())

M5.add(MaxPooling2D(pool_size=(2, 2)))

M5.add(Dropout(.2))

M5.add(Conv2D(64, (4, 4), activation='relu'))

M5.add(BatchNormalization())

M5.add(Conv2D(64, (4, 4), activation='relu'))

M5.add(BatchNormalization())

M5.add(MaxPooling2D(pool_size=(2, 2)))

M5.add(BatchNormalization())

M5.add(Dropout(.2))

M5.add(Flatten())

M5.add(Dense(1600,activation='relu'))
M5.add(Dense(800,activation='relu'))
M5.add(BatchNormalization())
M5.add(Dense(400,activation='relu'))
M5.add(Dense(210,activation='relu'))

M5.add(BatchNormalization())

M5.add(Dense(1, activation='sigmoid'))

M5.summary()
```


```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), activation='relu', input_
e))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, (4, 4), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(2000, activation='relu'))
model.add(Dense(1000, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(500, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.summary()


#COMPARING OUR DATA WITH THE TA MODEL
```

Persona 01

Models





Persona 01

Models

```
M6 = Sequential()

M6.add(Conv2D(32, kernel_size=(2, 2),strides=(1,1),activation='relu',input_shape=input_shape))

M6.add(BatchNormalization())

M6.add(Conv2D(32, (2, 2),strides=(2, 2), activation='relu'))

M6.add(BatchNormalization())

M6.add(MaxPooling2D(pool_size=(3, 3),strides=(2,2)))

M6.add(Dropout(.2))

M6.add(Conv2D(64, (3, 3), activation='relu'))

M6.add(BatchNormalization())

M6.add(Conv2D(64, (3, 3), activation='relu'))

M6.add(BatchNormalization())

M6.add(MaxPooling2D(pool_size=(2, 2)))

M6.add(Dropout(.2))

M6.add(Conv2D(64, (4, 4), activation='relu'))

M6.add(BatchNormalization())

M6.add(Conv2D(64, (4, 4), activation='relu'))

M6.add(BatchNormalization())

M6.add(MaxPooling2D(pool_size=(2, 2)))

M6.add(BatchNormalization())

M6.add(Dropout(.2))

M6.add(Flatten())

M6.add(Dense(1600,activation='relu'))
M6.add(Dense(800,activation='relu'))
M6.add(BatchNormalization())
M6.add(Dense(400,activation='relu'))
M6.add(Dense(210,activation='relu'))

M6.add(BatchNormalization())

M6.add(Dense(1, activation='sigmoid'))

M6.summary()
```


```
M5 = Sequential()

M5.add(Conv2D(32, kernel_size=(2, 2),strides=(1,1),activation='relu',input_shape=input_shape))

M5.add(BatchNormalization())

M5.add(Conv2D(32, (2, 2),strides=(2, 2), activation='relu'))

M5.add(BatchNormalization())

M5.add(MaxPooling2D(pool_size=(3, 3),strides=(2,2)))

M5.add(Dropout(.2))

M5.add(Conv2D(64, (3, 3), activation='relu'))

M5.add(BatchNormalization())

M5.add(Conv2D(64, (3, 3), activation='relu'))

M5.add(BatchNormalization())

M5.add(MaxPooling2D(pool_size=(2, 2)))

M5.add(Dropout(.2))

M5.add(Conv2D(64, (4, 4), activation='relu'))

M5.add(BatchNormalization())

M5.add(Conv2D(64, (4, 4), activation='relu'))

M5.add(BatchNormalization())

M5.add(MaxPooling2D(pool_size=(2, 2)))

M5.add(BatchNormalization())

M5.add(Dropout(.2))

M5.add(Flatten())

M5.add(Dense(1600,activation='relu'))
M5.add(Dense(800,activation='relu'))
M5.add(BatchNormalization())
M5.add(Dense(400,activation='relu'))
M5.add(Dense(210,activation='relu'))

M5.add(BatchNormalization())

M5.add(Dense(1, activation='sigmoid'))

M5.summary()
```

Persona 01

Models

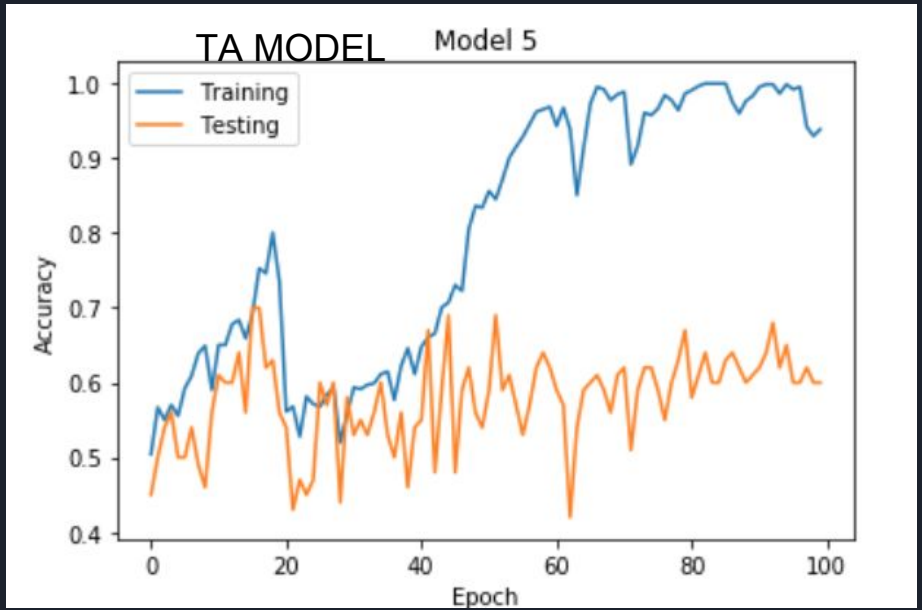

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), activation='relu', input_shape=input_shape))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, (4, 4), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(2000, activation='relu'))
model.add(Dense(1000, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(500, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.summary()


#COMPARING OUR DATA WITH THE TA MODEL
```

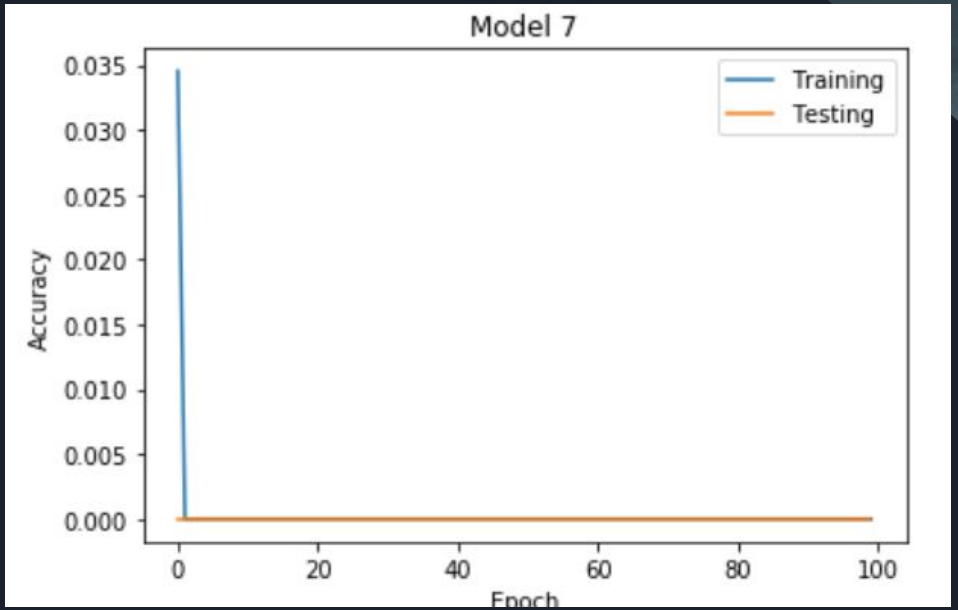
```
M7 = Sequential()
M7.add(Dense(1200, input_shape=input_shape, activation='sigmoid'))
M7.add(Dense(600, activation='sigmoid', name='hidden'))
M7.add(Flatten())
M7.add(Dense(1, activation='sigmoid', name='output'))
```

Persona 01

Models

TA MODEL





Persona 01

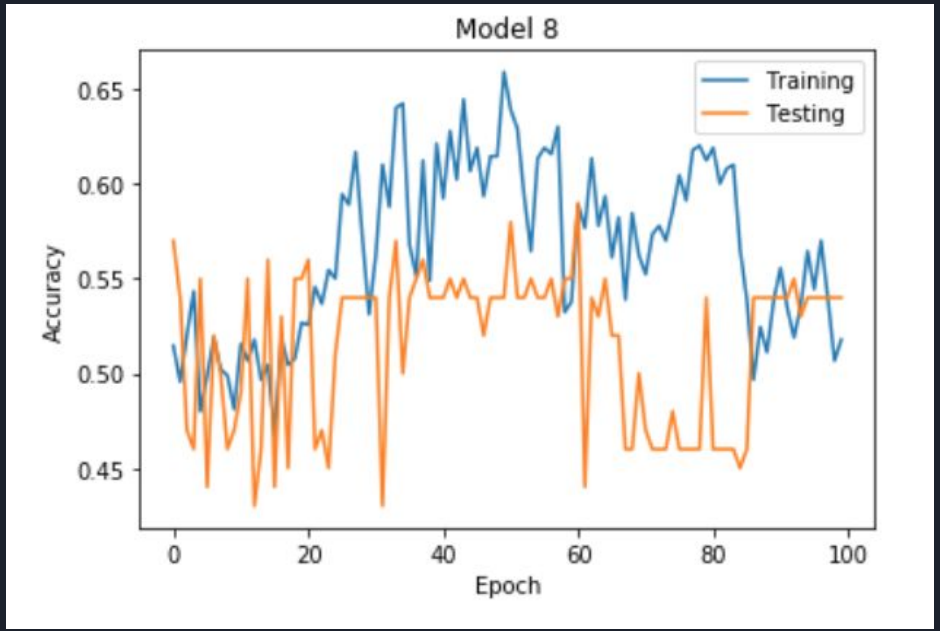
Models

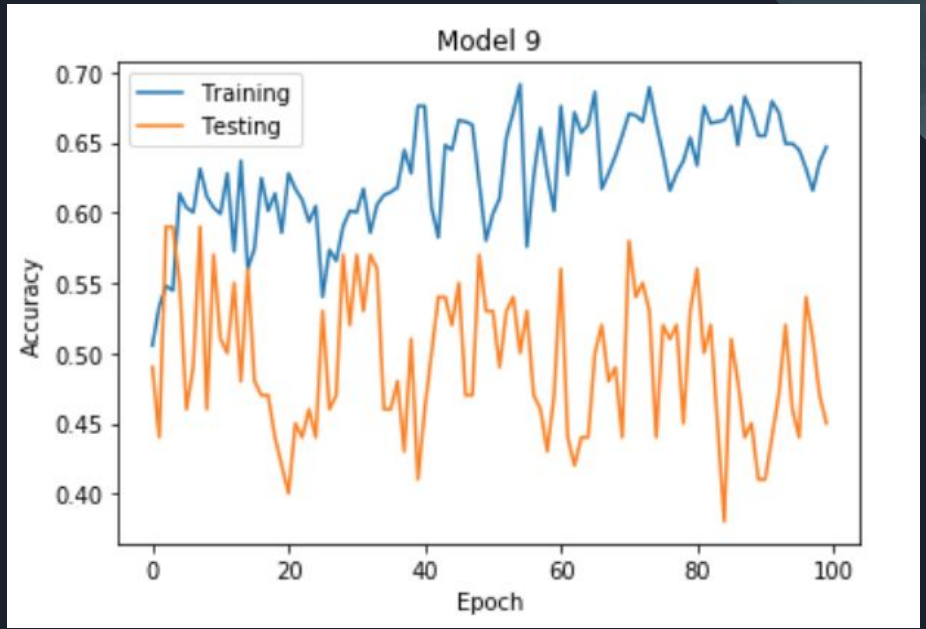
```
M8 = Sequential()
M8.add(Conv2D(32, kernel_size=(2, 2),strides=(1,1),activation='relu',input_shape=input_shape))
M8.add(BatchNormalization())
M8.add(Conv2D(32, (2, 2),strides=(2, 2), activation='relu'))
M8.add(BatchNormalization())
M8.add(MaxPooling2D(pool_size=(3, 3),strides=(2,2)))
M8.add(Dropout(.2))
M8.add(Conv2D(64, (3, 3), activation='relu'))
M8.add(BatchNormalization())
M8.add(Conv2D(64, (3, 3), activation='relu'))
M8.add(BatchNormalization())
M8.add(MaxPooling2D(pool_size=(2, 2)))
M8.add(Dropout(.2))
M8.add(Conv2D(64, (4, 4), activation='relu'))
M8.add(BatchNormalization())
M8.add(Conv2D(64, (4, 4), activation='relu'))
M8.add(BatchNormalization())
M8.add(MaxPooling2D(pool_size=(2, 2)))
M8.add(BatchNormalization())
M8.add(Dropout(.2))
M8.add(Flatten())
M8.add(Dense(1600,activation='sigmoid'))
M8.add(Dense(800,activation='sigmoid'))
M8.add(BatchNormalization())
M8.add(Dense(400,activation='sigmoid'))
M8.add(Dense(210,activation='sigmoid'))
M8.add(BatchNormalization())
M8.add(Dense(1, activation='sigmoid'))
M8.summary()
```


```
M9 = Sequential()
M9.add(Conv2D(32, kernel_size=(2, 2),strides=(1,1),activation='relu',input_shape=input_shape))
M9.add(BatchNormalization())
M9.add(Conv2D(32, (2, 2),strides=(2, 2), activation='relu'))
M9.add(BatchNormalization())
M9.add(MaxPooling2D(pool_size=(3, 3),strides=(2,2)))
M9.add(Dropout(.2))
M9.add(Flatten())
M9.add(Dense(1600,activation='sigmoid'))
M9.add(BatchNormalization())
M9.add(Dense(1, activation='sigmoid'))
M9.summary()
```

Persona 01

Models





Persona 01

Conclusion:

The model that was the most successful was a range between Model 6 and Model 8. However, the features that 100% defines what a Hot-Dog is,is still among the mist. The technology is close, but not there. The question still remains; What exactly makes a Hot-Dog a Hot-Dog?