TensorFlow

CNN with TF

Rishit Dagli

10 STD, TEDX, Ted-Ed speaker|Google certified mobile site developer|Intel AI Scholar|2XGCP Champ|Mozilla Mumbai Lead

rishit.tech

http://bit.ly/tfugm

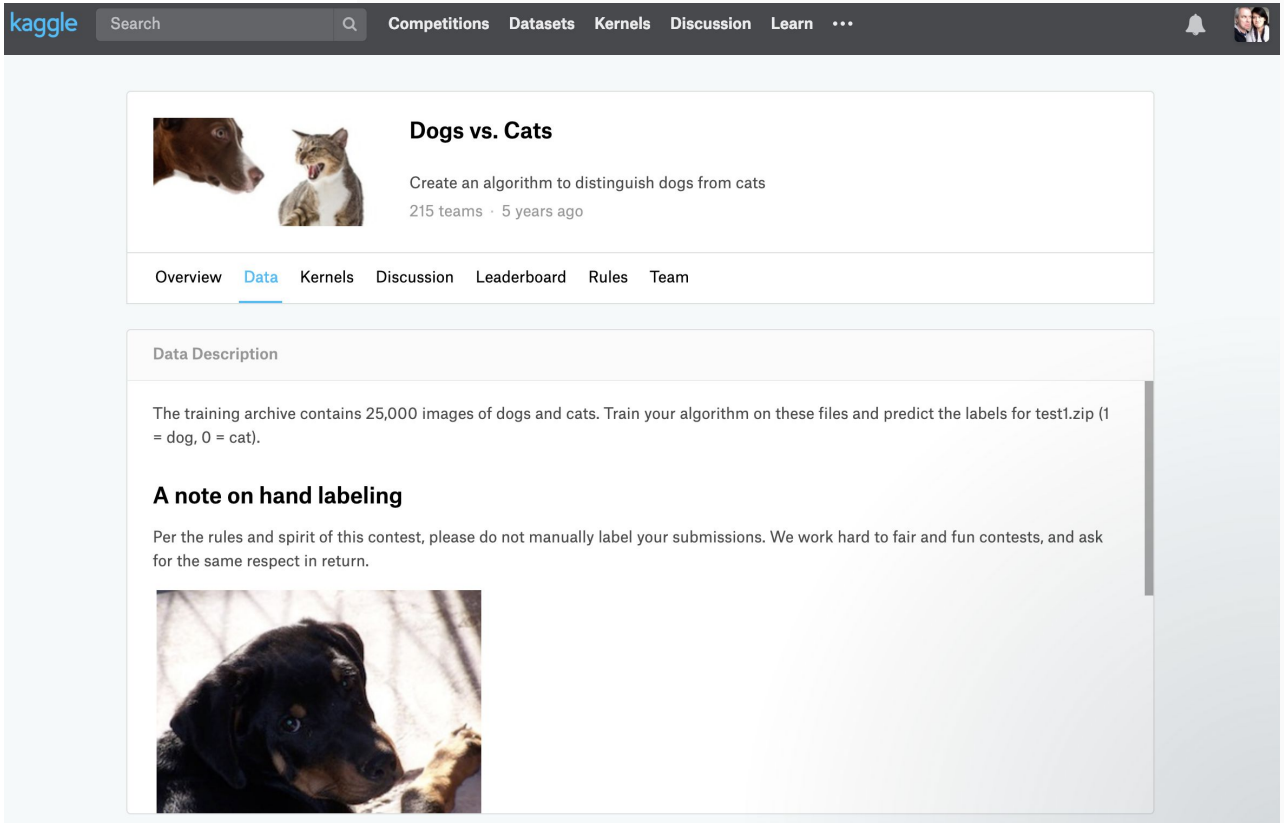
http://bit.ly/tfugm-p

People with no idea about AI,
Wondering AI will destroy humanity

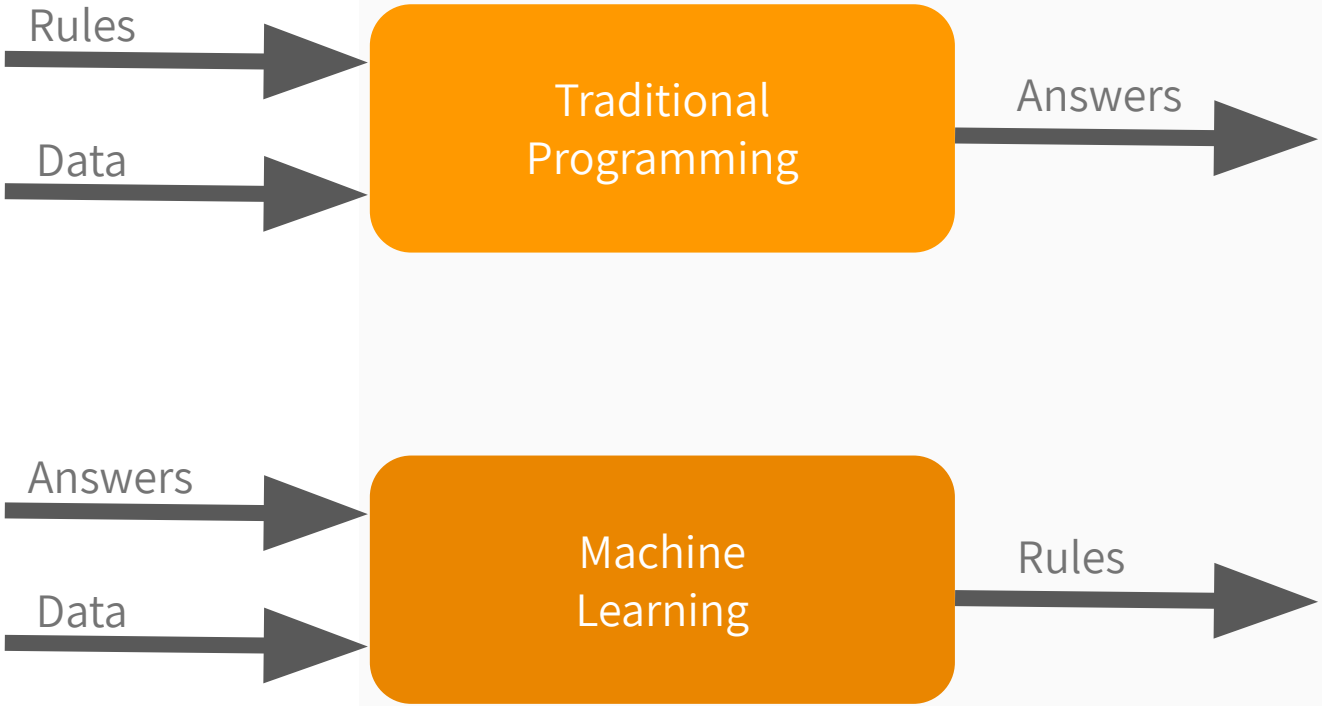
Me wondering why my NN is classifying a cat as a dog

rishit.tech

https://www.kaggle.com/c/dogs-vs-cats/data




Rules
Data
Traditional Programming
Answers


Rules
Data
Traditional Programming
Answers
Answers
Data
Machine Learning
Rules

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

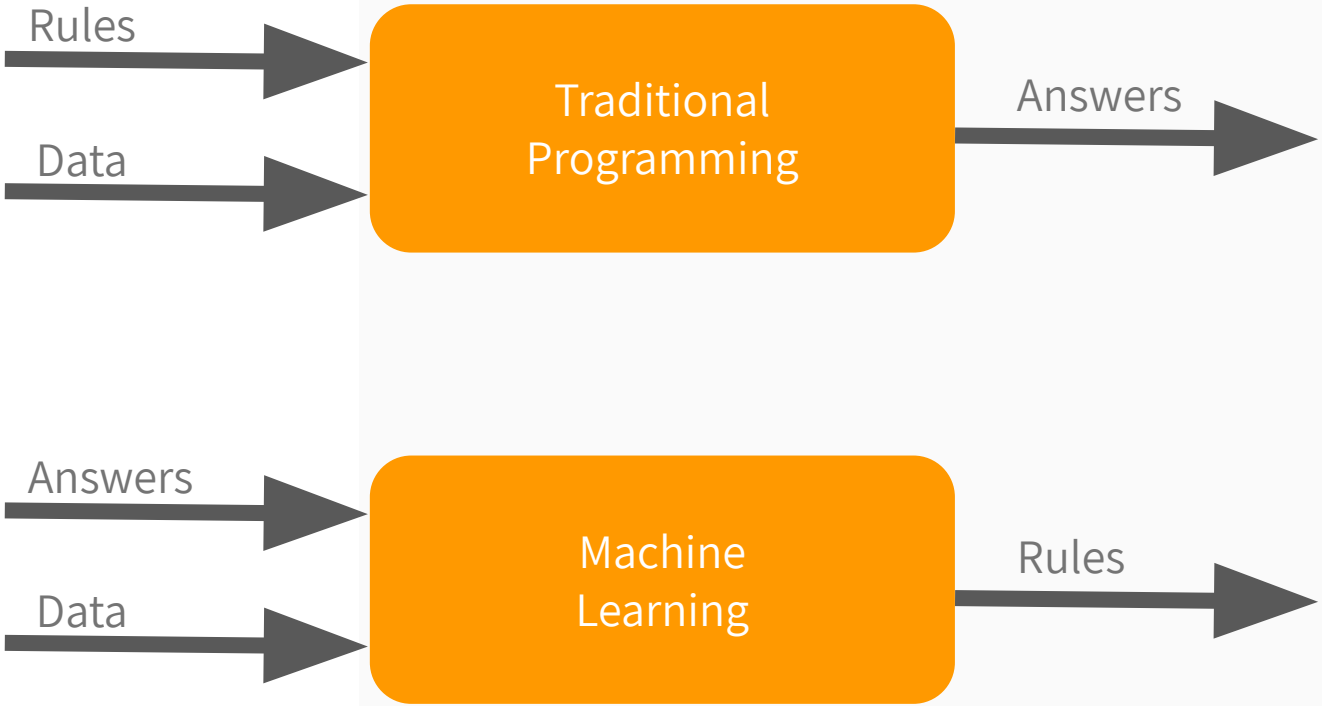
Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

```
// Oh crap
```


Rules
Data
Traditional Programming
Answers
Answers
Data
Machine Learning
Rules

Activity Recognition

```
0101001010100101010
1001010101001011101
0100101010010101001
0101001010100101010
```

Label = WALKING

```
1010100101001010101
0101010010010010001
0010011111010101111
1010100100111101011
```

Label = RUNNING

```
1001010011111010101
1101010111010101110
1010101111010101011
1111110001111010101
```

Label = BIKING

```
1111111111010011101
0011111010111110101
0101110101010101110
1010101010100111110
```

Label = GOLFING
(Sort of)

What to expect???

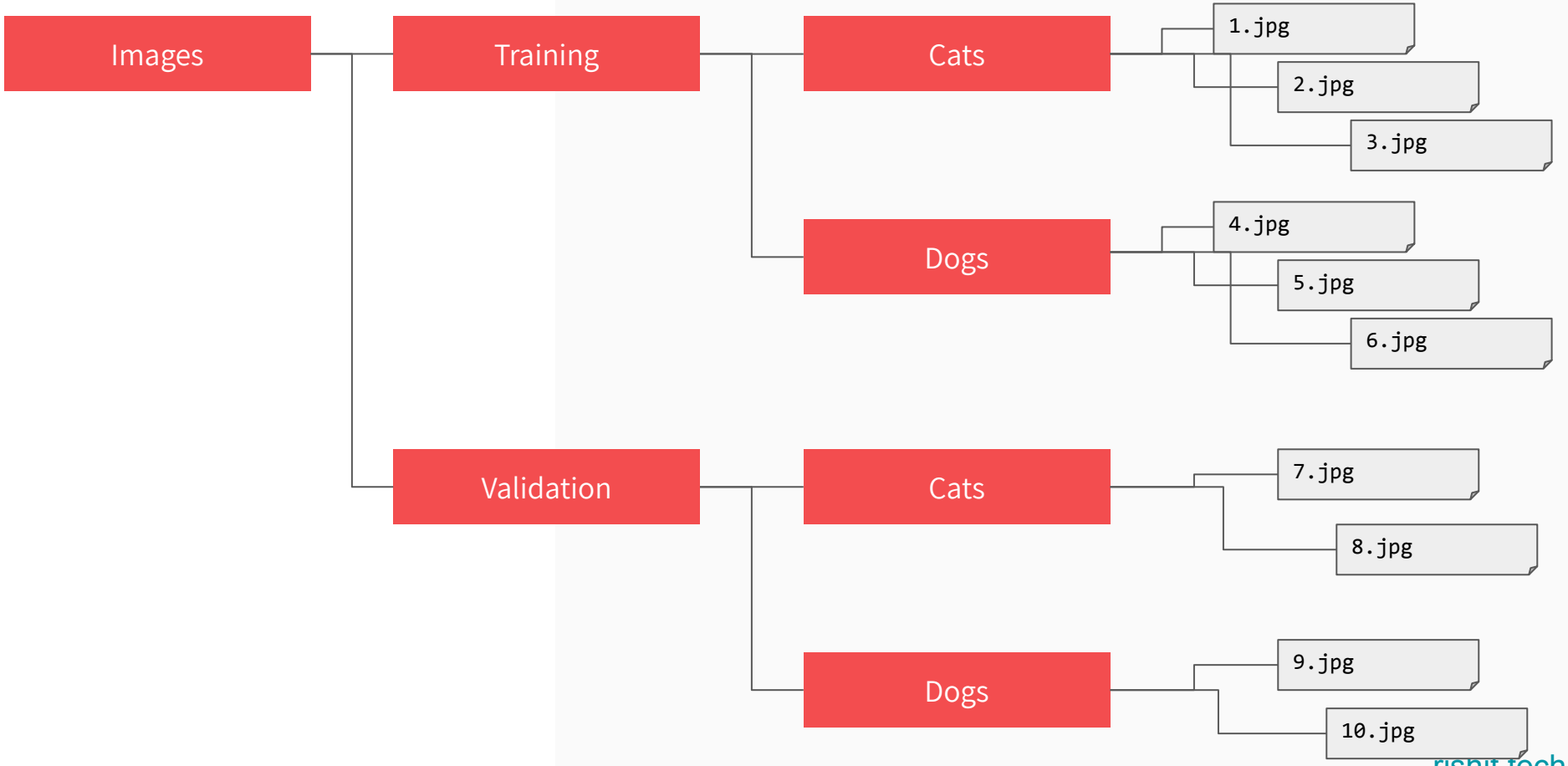
- Building *data input pipelines* using the tf.keras.preprocessing.image.ImageDataGenerator class to efficiently work with data on disk to use with the model.

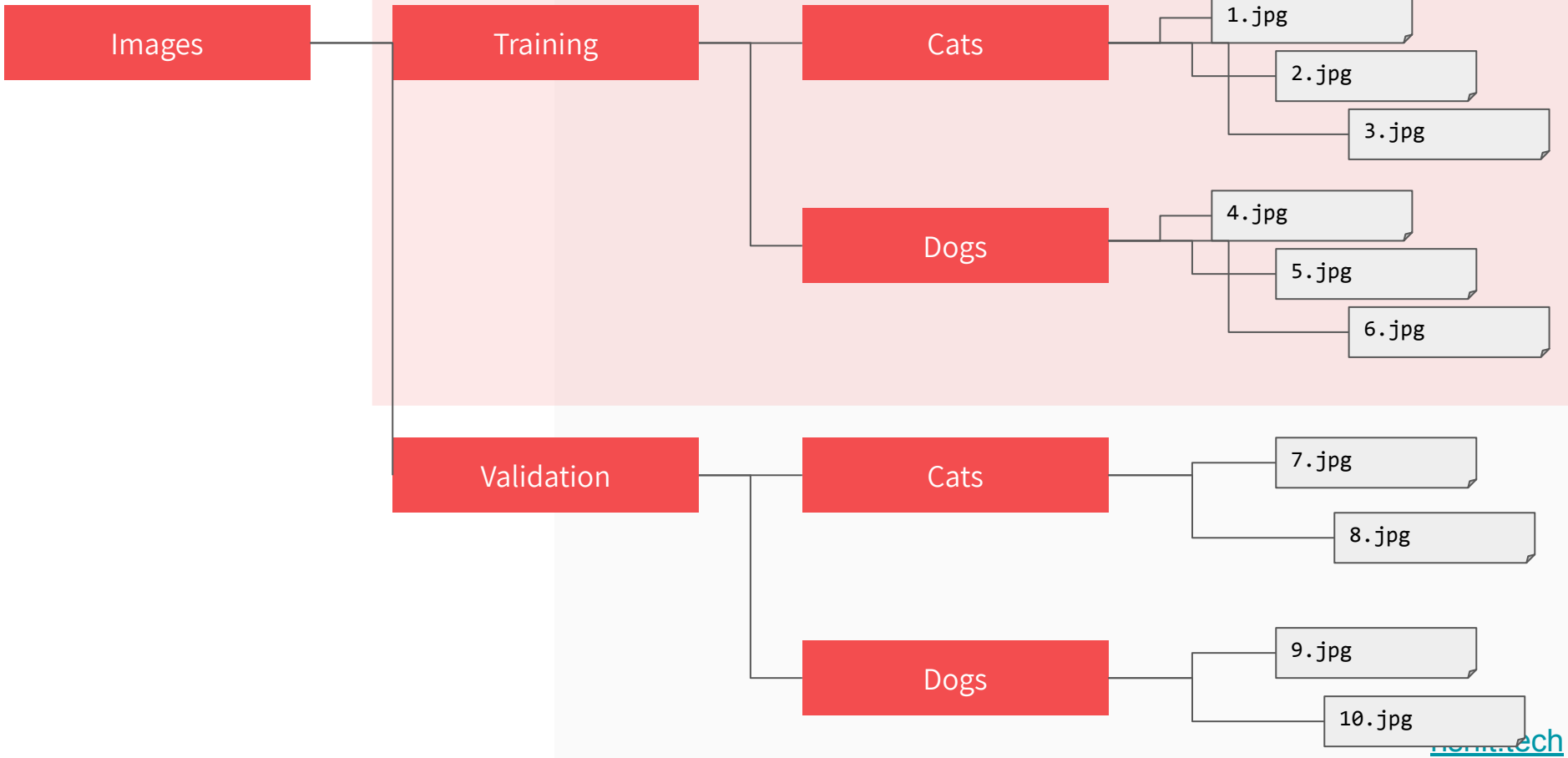
- Build and test model.

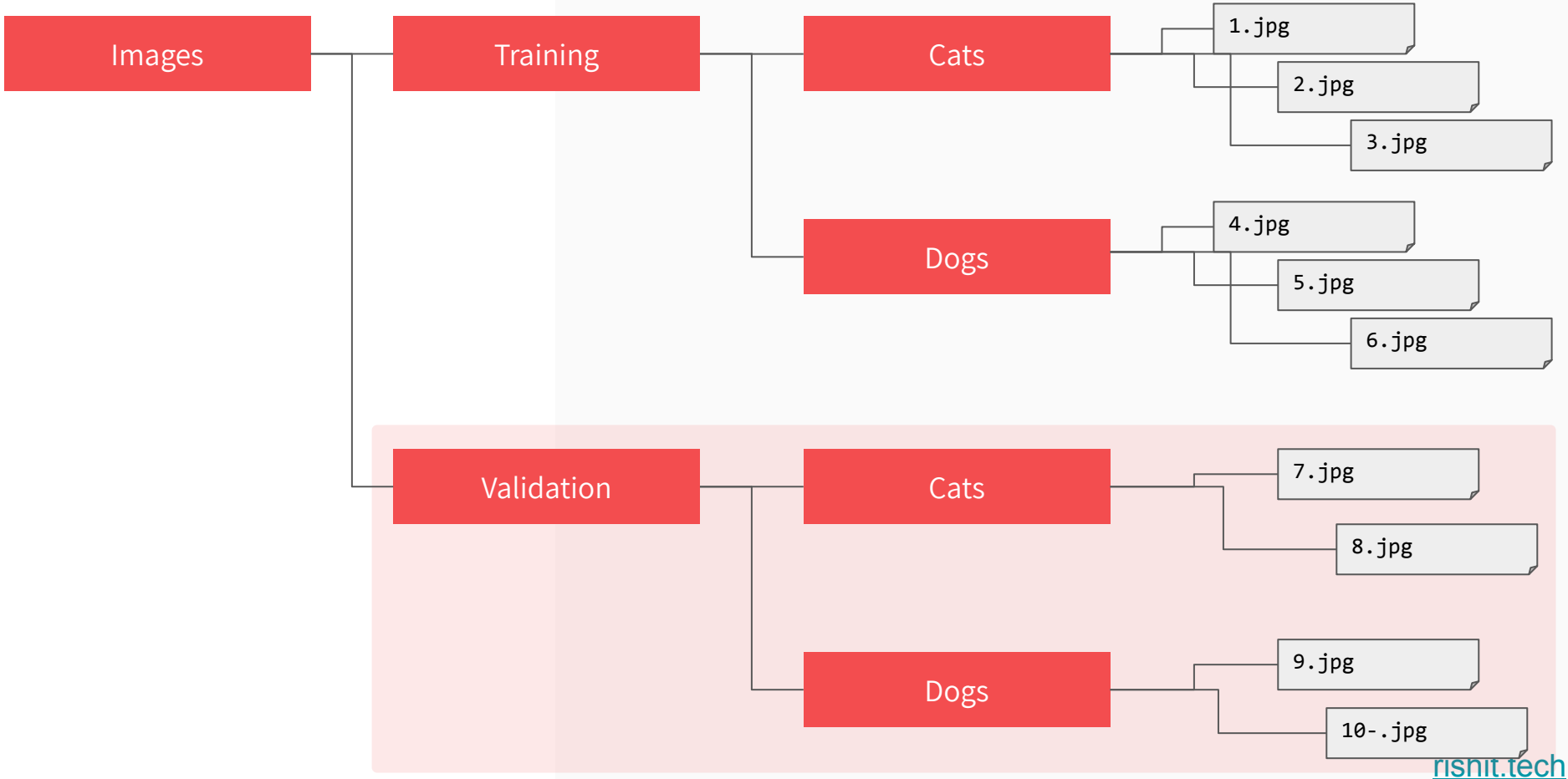
- *Overfitting* —How to identify and prevent it.

- *Data augmentation* and *dropout* —techniques to fight overfitting

Understanding the Data


Images
Training
Cats
1.jpg
2.jpg
3.jpg
Dogs
4.jpg
5.jpg
6.jpg
Validation
Cats
7.jpg
8.jpg
Dogs
9.jpg
10.jpg


Images
Training
Cats
1.jpg
2.jpg
3.jpg
Dogs
4.jpg
5.jpg
6.jpg
Validation
Cats
7.jpg
8.jpg
Dogs
9.jpg
10.jpg


Images
Training
Cats
1.jpg
2.jpg
3.jpg
Dogs
4.jpg
5.jpg
6.jpg
Validation
Cats
7.jpg
8.jpg
Dogs
9.jpg
10-.jpg

```
total training cat images: 1000
total training dog images: 1000
total validation cat images: 500
total validation dog images: 500
--
Total training images: 2000
Total validation images: 1000
```

Imports

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers
import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image
import  ImageDataGenerator

import os
import numpy as np
import matplotlib.pyplot as plt
```

Parameters

```
batch_size = 128
epochs = 15
IMG_HEIGHT = 150
IMG_WIDTH = 150
```

Data Preparation

1. Read images from the disk.
2. Decode contents of these images and convert it into proper grid format as per their RGB content.
3. Convert them into floating point tensors.
4. Rescale the tensors from values between 0 and 255 to values between 0 and 1, as neural networks prefer to deal with small input values.

```
from tensorflow.keras.preprocessing.image
import ImageDataGenerator
```

```
# Generator for our training data
train_image_generator = ImageDataGenerator(rescale=1./255)

train_data_gen =  train_iamge_generator.flow_from_directory(
                    batch_size=batch_size,
                    directory=train_dir,
                    shuffle=True,
                    target_size=(IMG_HEIGHT,IMG_WIDTH)
                    class_mode='binary')
```

```
# Generator for our training data
train_image_generator = ImageDataGenerator(rescale=1./255)

train_data_gen = train_image_generator.flow_from_directory(
                    batch_size=batch_size,
                    directory=train_dir,
                    shuffle=True,
                    target_size=(IMG_HEIGHT,IMG_WIDTH)
                    class_mode='binary')
```

```
# Generator for our training data
train_image_generator = ImageDataGenerator(rescale=1./255)

train_data_gen = train_image_generator.flow_from_directory(
                    batch_size=batch_size,
                    directory=train_dir,
                    shuffle=True,
                    target_size=(IMG_HEIGHT,IMG_WIDTH)
                    class_mode='binary')
```

```
# Generator for our training data
train_image_generator = ImageDataGenerator(rescale=1./255)

train_data_gen = train_image_generator.flow_from_directory(
                    batch_size=batch_size,
                    directory= train_dir,
                    shuffle=True,
                    target_size=(IMG_HEIGHT,IMG_WIDTH)
                    class_mode='binary')
```

```
# Generator for our training data
train_image_generator = ImageDataGenerator(rescale=1./255)

train_data_gen = train_image_generator.flow_from_directory(
                    batch_size=batch_size,
                    directory= train_dir,
                    shuffle=True,
                    target_size=(IMG_HEIGHT,IMG_WIDTH)
                    class_mode='binary')
```

```
# Generator for our training data
train_image_generator = ImageDataGenerator(rescale=1./255)

train_data_gen = train_image_generator.flow_from_directory(
                    batch_size=batch_size,
                    directory= train_dir,
                    shuffle=True,
                    target_size=(IMG_HEIGHT,IMG_WIDTH)
                    class_mode='binary')
```

```
# Generator for our validation data
validation_image_generator = ImageDataGenerator(rescale=1./255)

val_data_gen = validation_imadata_generator.flow_from_directory
                  (
                      batch_size=batch_size,
                      directory= validation_dir,
                      shuffle=True,
                      target_size=(IMG_HEIGHT,IMG_WIDTH)
                      class_mode='binary')
```

TensorFlow

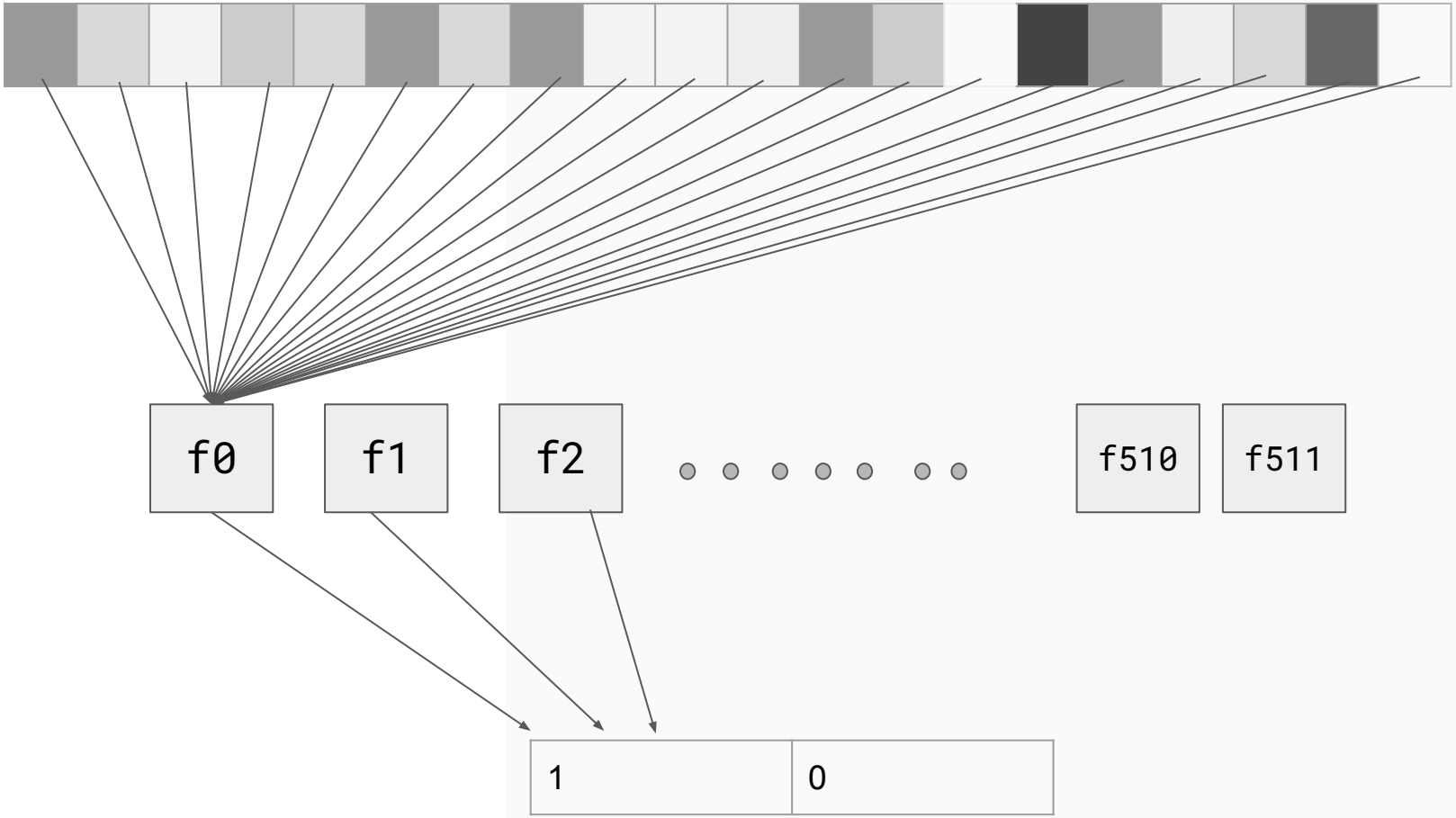
Model

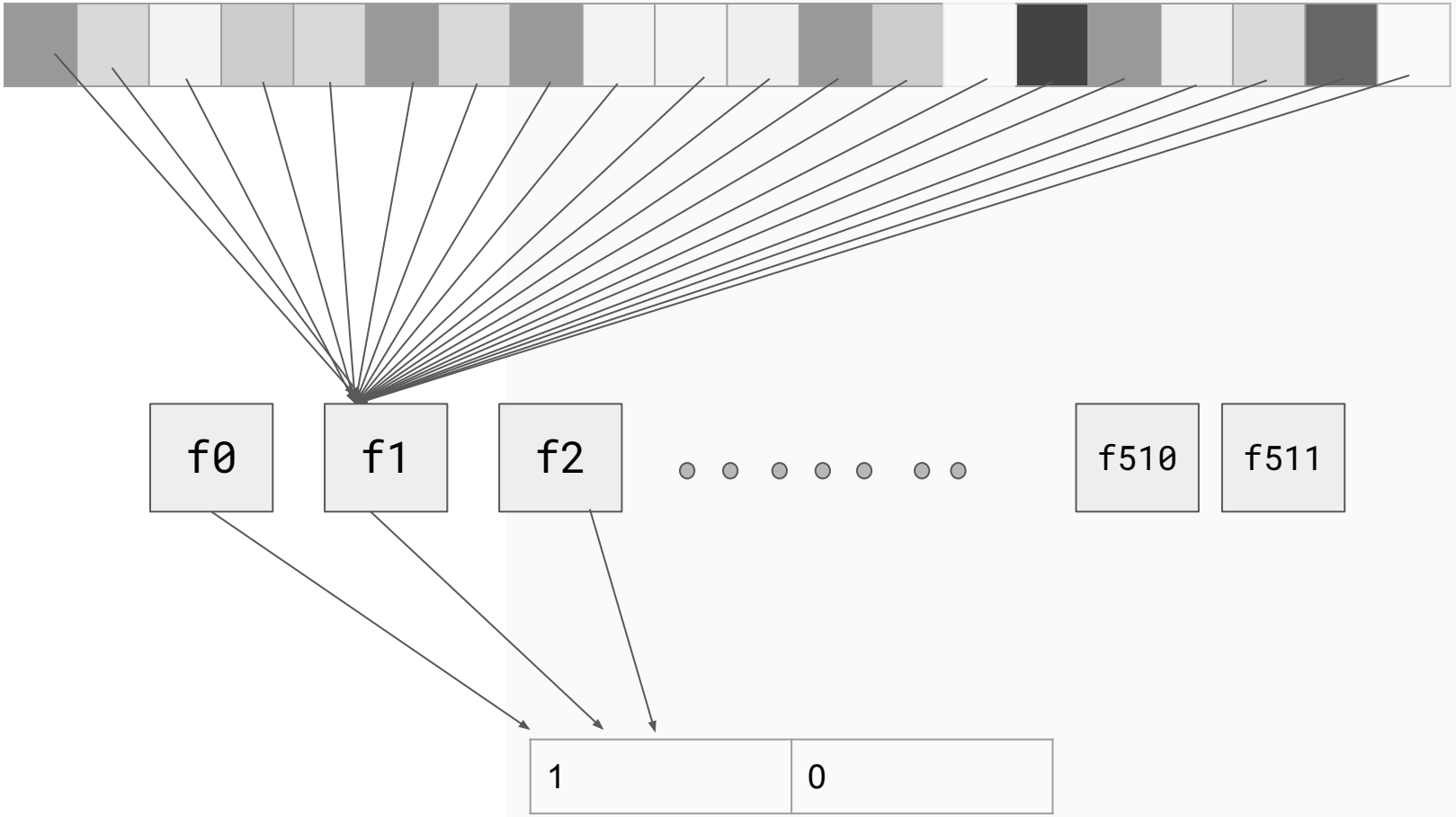
Let’s understand and Build a Convolutional Neural Network

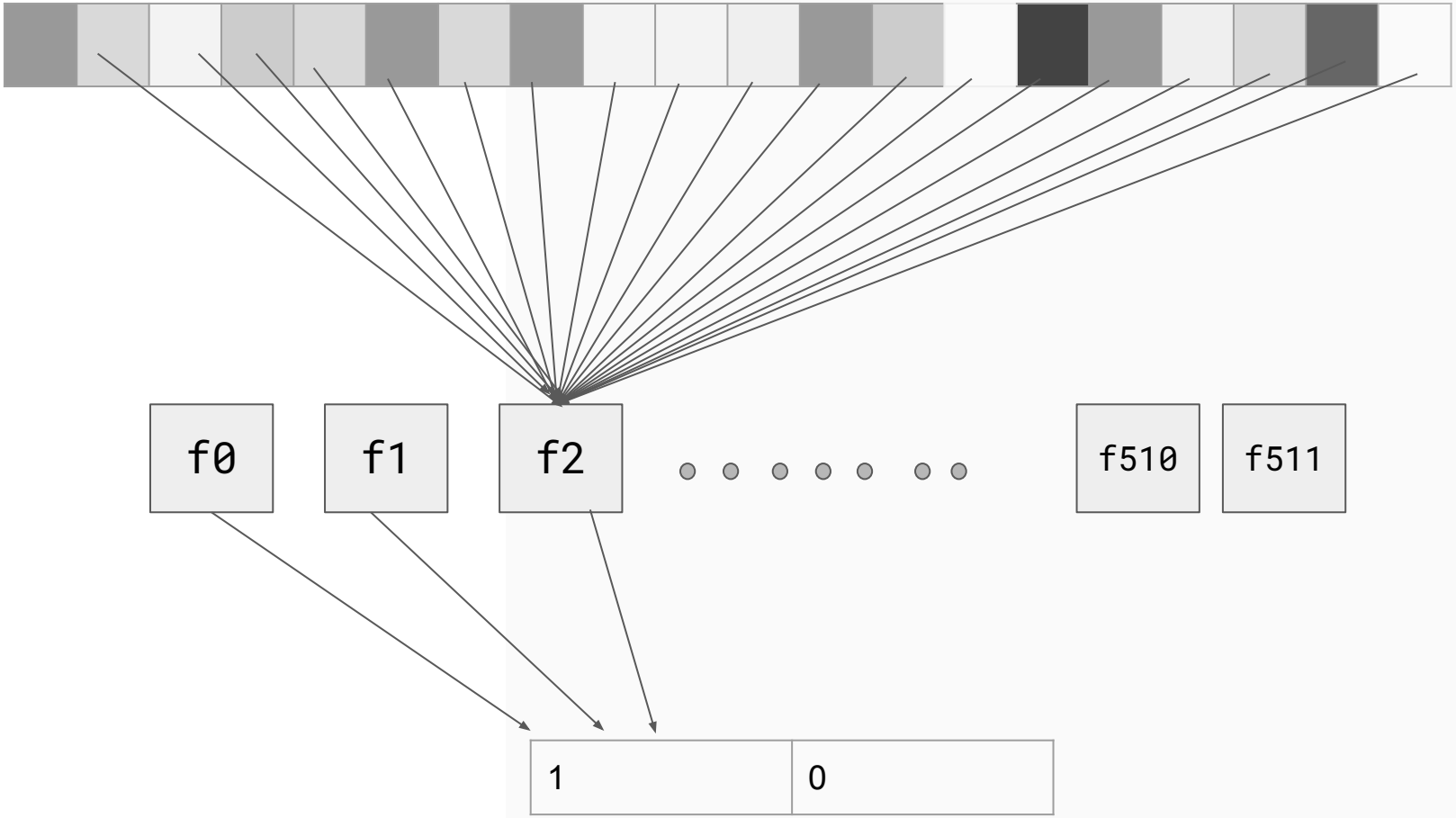
```
model = Sequential([
    Flatten( input_shape = (150,150,3),
    Dense(512 , activation = ‘relu’,
    Dense(2, activation=’softmax’)
])
```

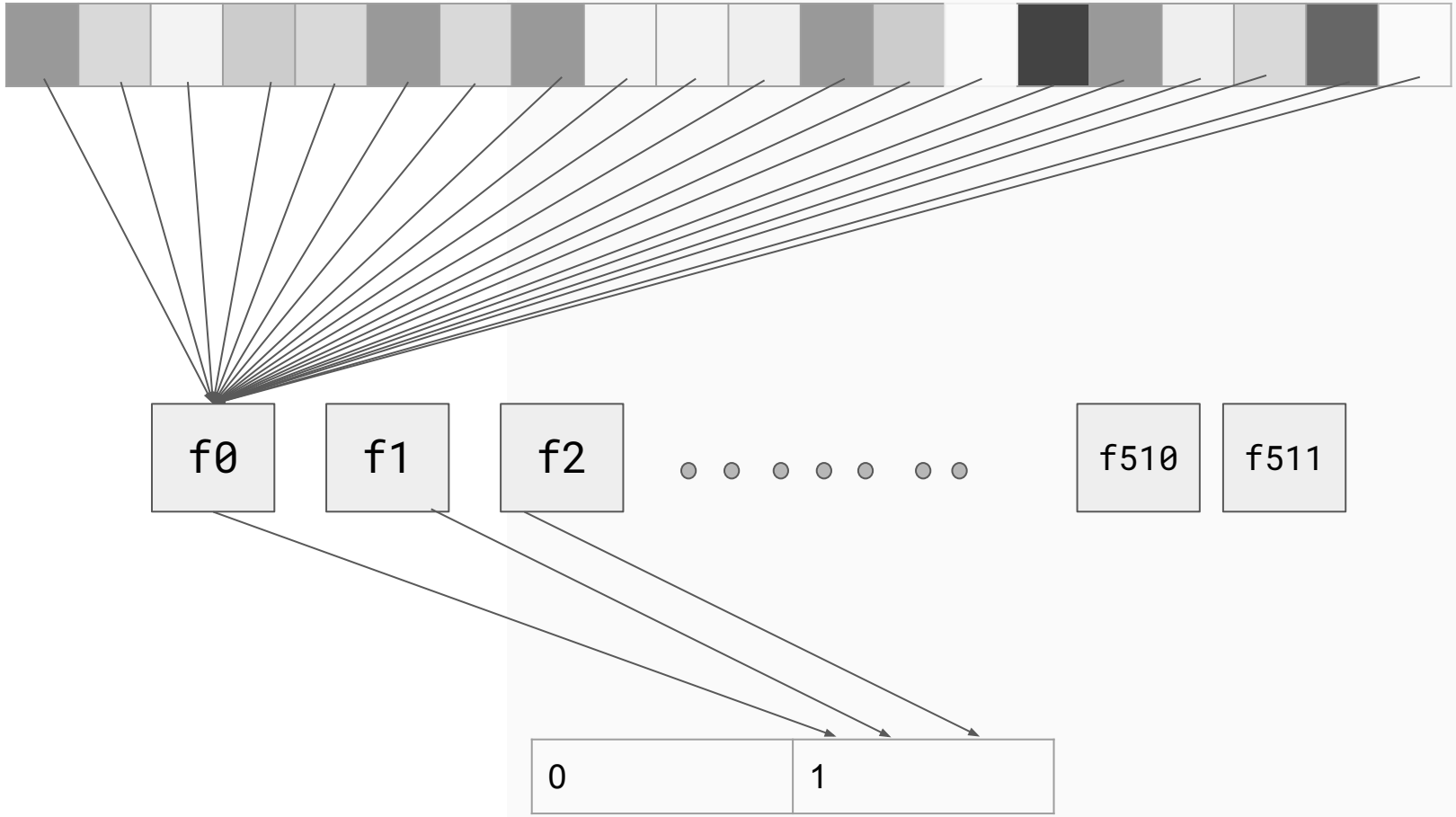
```
model = Sequential([
    Flatten( input_shape = (150,150,3),
    Dense(512 , activation = 'relu',
    Dense( 2 , activation='softmax')
])
```

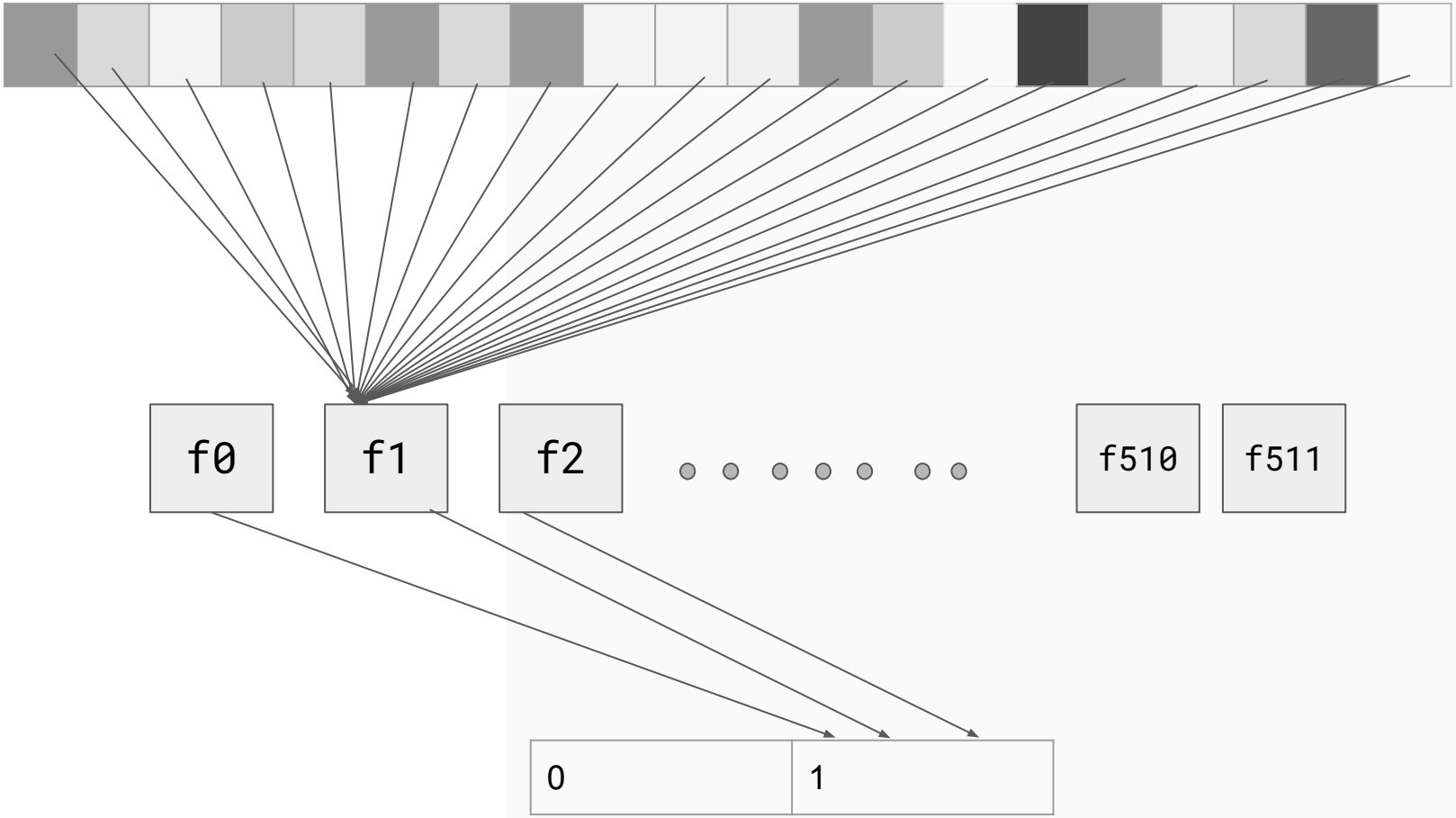
```
model = Sequential([
    Flatten( input_shape = (150,150,3),
    Dense( 512 , activation = ‘relu’,
    Dense(2, activation=’softmax’)
])
```

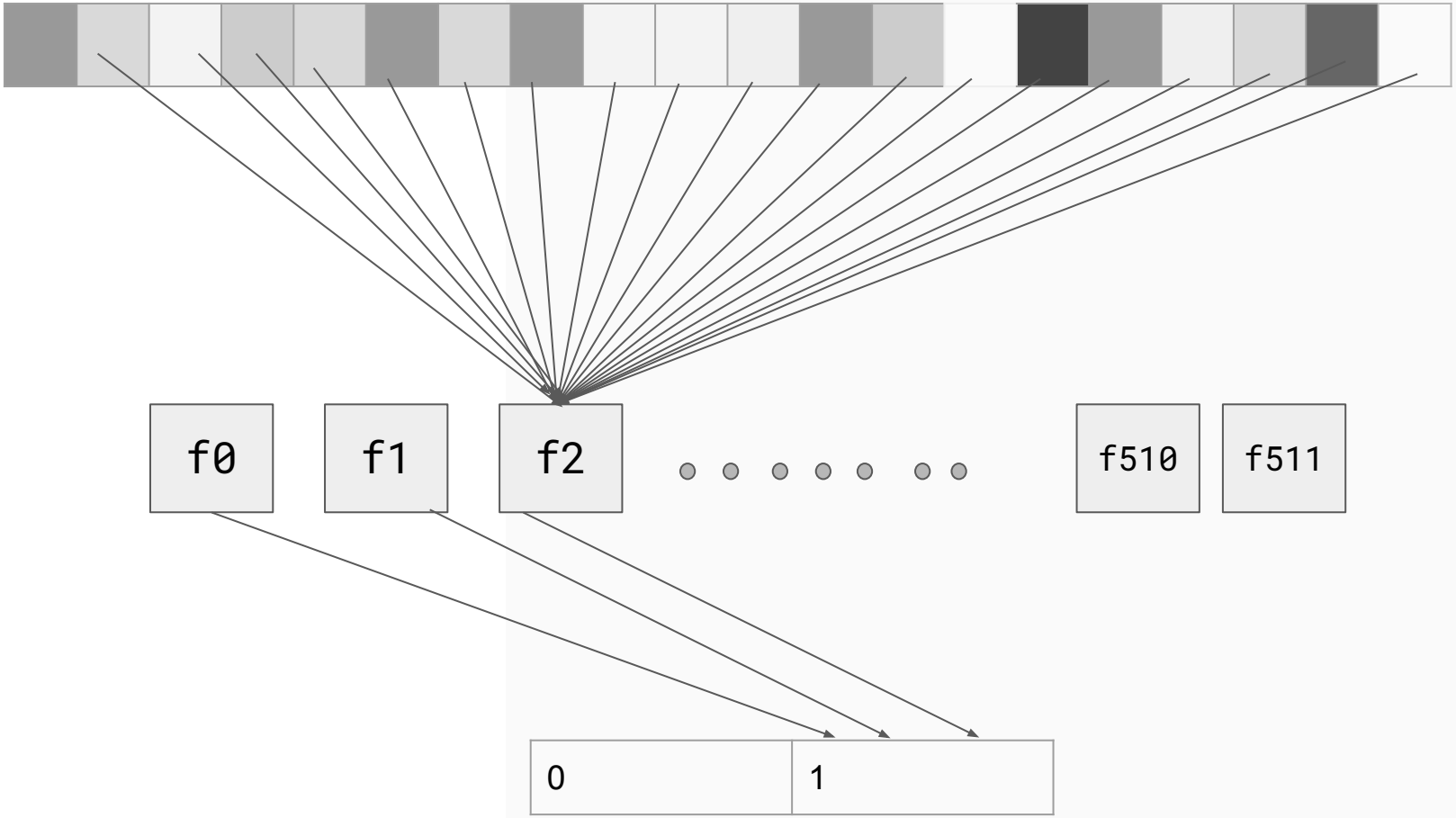

f0
f1
f2
f510
f511
1
0


f0
f1
f2
f510
f511
1
0


f0
f1
f2
f510
f511
1
0

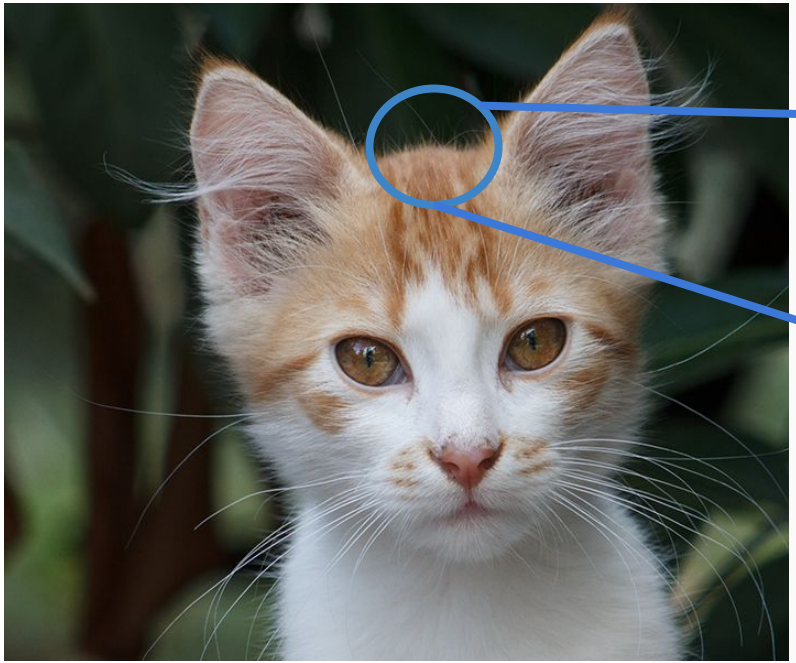

f0
f1
f2
f510
f511
0
1


f0
f1
f2
f510
f511
0
1


f0
f1
f2
f510
f511
0
1

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
             )

 model.fit(....., epochs = 100)
```



0	64	128
48	192	144
142	226	168

Current Pixel Value is 192

Consider neighbor Values

-1	0	-2
.5	4.5	-1.5
1.5	2	-3

Filter Definition

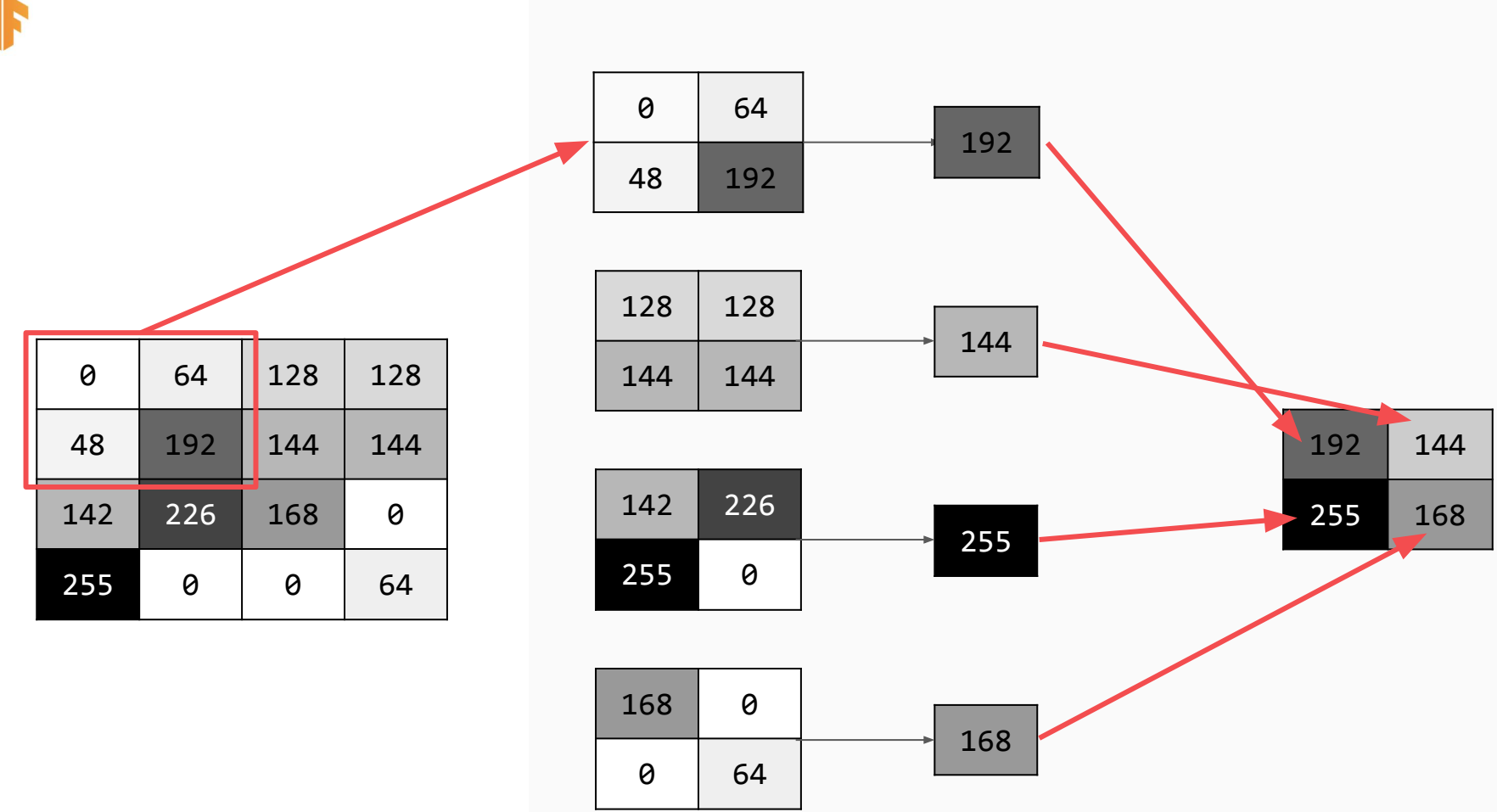
```
CURRENT_PIXEL_VALUE = 192
NEW_PIXEL_VALUE = (-1 * 0) + (0 * 64) + (-2 * 128) +
                  (.5 * 48) + (4.5 * 192) + (-1.5 * 144) +
                  (1.5 * 42) + (2 * 226) + (-3 * 168)
```

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

```
model = Sequential([
        Conv2D(16, (3,3), padding='same', activation='relu',
        input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(2,2),
    Conv2D(32, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

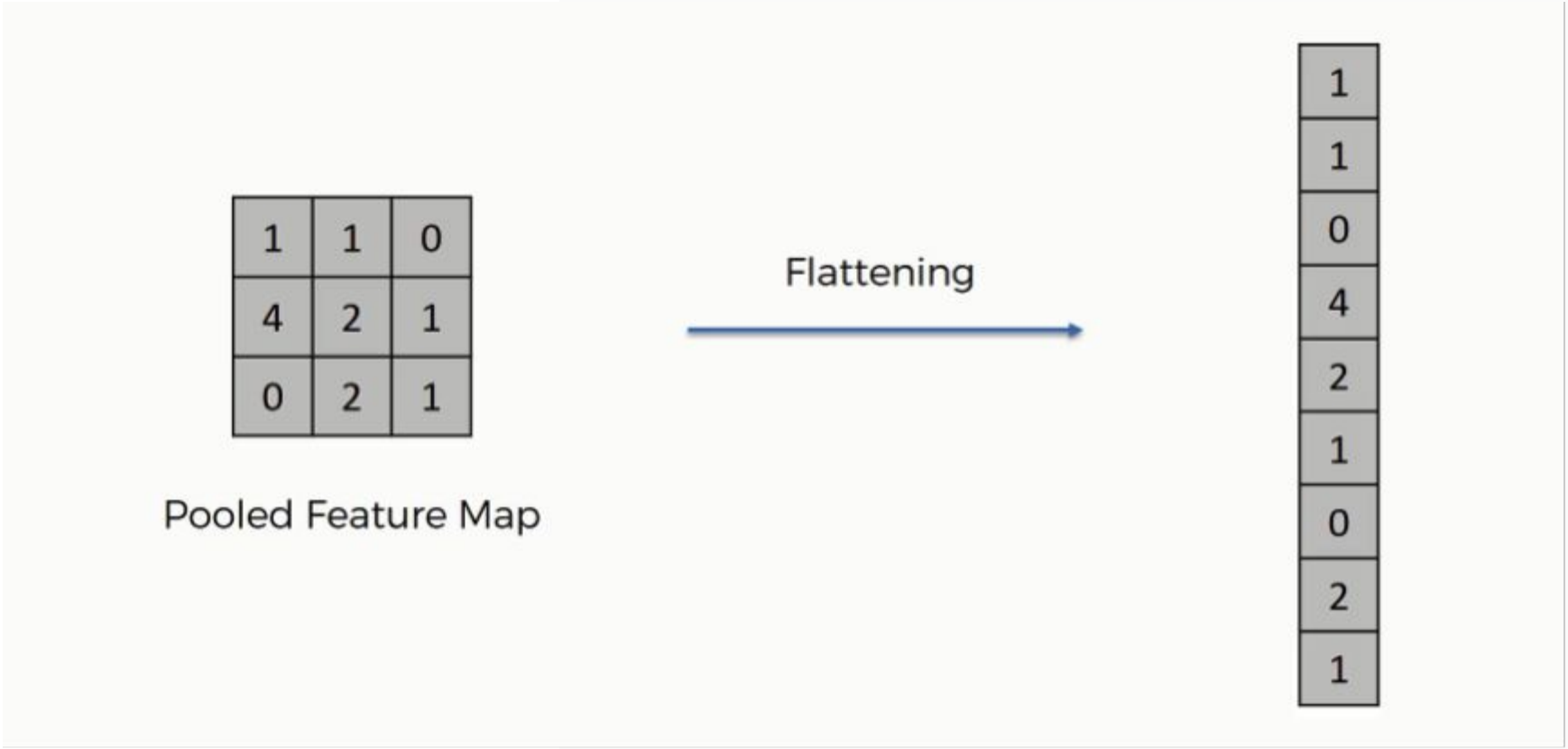
```
model = Sequential([
        Conv2D(16, (3,3), padding='same', activation='relu',
        input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(2,2),
    Conv2D(32, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])
```


0 64 128 128
48 192 144 144
142 226 168 0
255 0 0 64
0 64
48 192
192
128 128
144 144
144
142 226
255 0
255
168 0
0 64
168
192 144
255 168

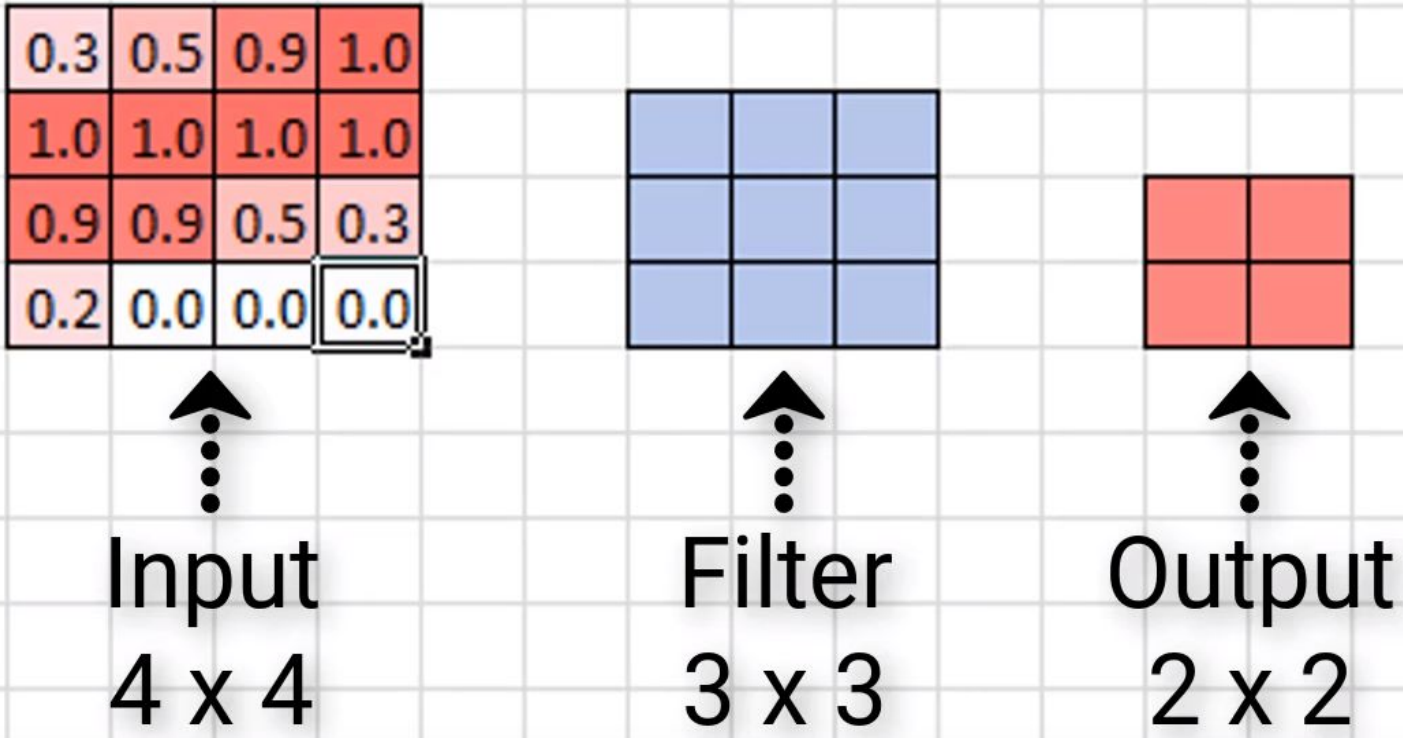
Max Pooling Example

Max Pooling 2X2

```
model = Sequential([
        Conv2D(16, (3,3), padding='same', activation='relu',
        input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(2,2),
    Conv2D(32, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])
```


1 1 0
4 2 1
0 2 1
Pooled Feature Map
Flattening
1
1
0
4
2
1
0
2
1

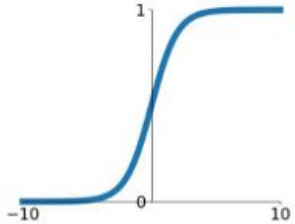
```
model = Sequential([
        Conv2D(16, (3,3), padding='same', activation='relu',
        input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(2,2),
    Conv2D(32, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])
```


0.3 0.5 0.9 1.0
1.0 1.0 1.0 1.0
0.9 0.9 0.5 0.3
0.2 0.0 0.0 0.0
Input
4 x 4
Filter
3 x 3
Output
2 x 2

```
model = Sequential([
        Conv2D(16, (3,3), padding='same', activation='relu',
        input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(2,2),
    Conv2D(32, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

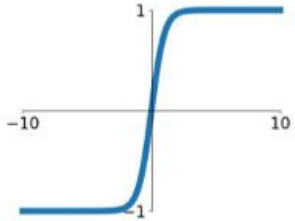
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



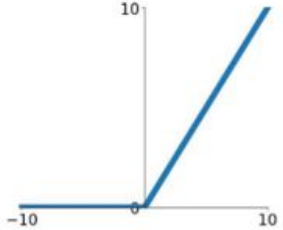
tanh

$$\tanh(x)$$



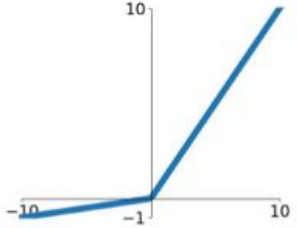
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

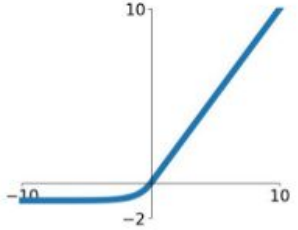


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

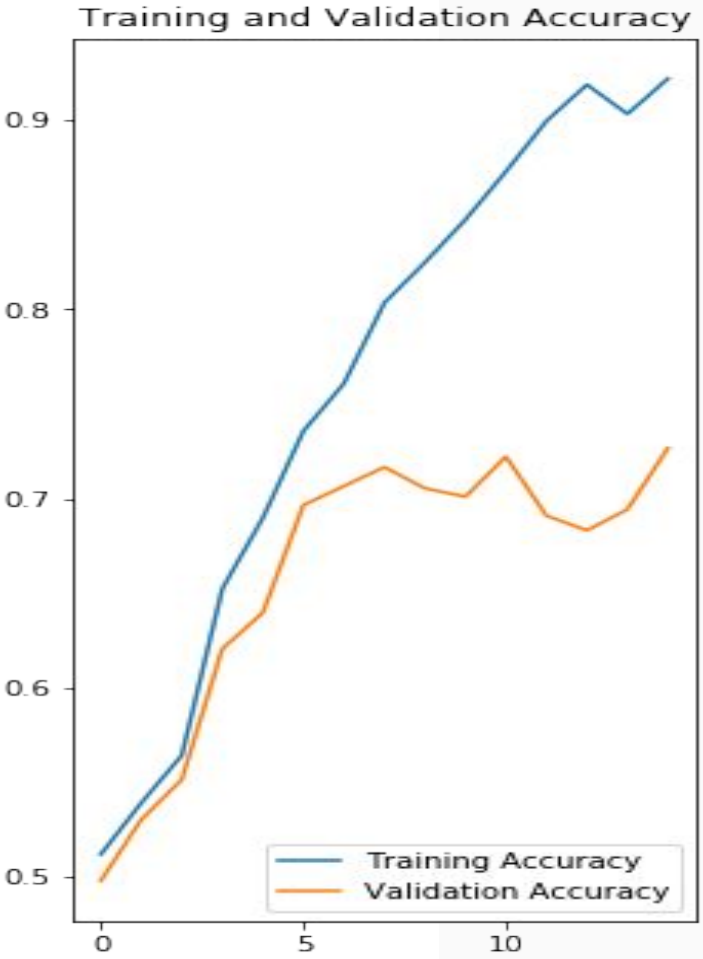
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

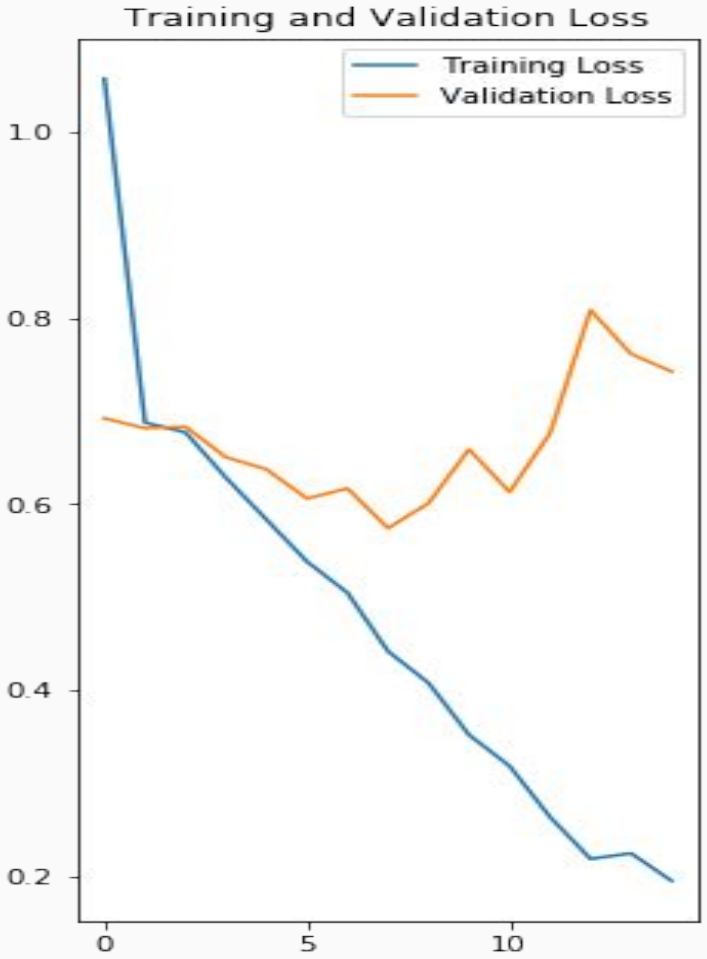


```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 150, 150, 16)      448
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 75, 75, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 75, 75, 32)        4640
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 37, 37, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 18, 18, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 20736)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               10617344
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 10,641,441
Trainable params: 10,641,441
Non-trainable params: 0
```

```
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=val_data_gen,
    validation_steps=total_val // batch_size
)
```


Training and Validation Accuracy
0.9
0.8
0.7
0.6
0.5
0
5
10
Training Accuracy
Validation Accuracy


Training and Validation Loss
1.0
0.8
0.6
0.4
0.2
0
5
10
Training Loss
Validation Loss

Overfitting

When there are small number of training examples, the model sometimes learns from noises or unwanted details from the training examples- to an extent it negatively impacts the performance of the model on new examples. This is what know as overfitting

How Do You avoid it??

1. Add more data
2. Use data augmentation
3. Use architectures that generalize well.

Data Augmentation

Apply some random transformation.

The Goal is the model never sees the exact same picture twice.

Use ImageDataGenerator to perform transformation

Horizontal Flip

```
image_gen = ImageDataGenerator(rescale=1./255,
horizontal_flip=True)
```

Rotate the Image

```
image_gen = ImageDataGenerator(rescale=1./255,
rotation_range=45)
```

Zoom Augmentation

```
image_gen = ImageDataGenerator(rescale=1./255,
zoom_range=0.5)
```

Let’s Put this together.

```
image_gen_train = ImageDataGenerator(
                    rescale=1./255,
                    rotation_range=45,
                    width_shift_range=.15,
                    height_shift_range=.15,
                    horizontal_flip=True,
                    zoom_range=0.5
                    )
```

```
train_data_gen =  image_gen_train.flow_from_directory(
                    batch_size=batch_size,
                    directory=train_dir,
                    shuffle=True,
                    target_size=(IMG_HEIGHT,IMG_WIDTH)
                    class_mode='binary')
```

DropOut

Another technique to overcome overfitting.

It drops some of the output units from the applied layer during the training process

It takes values such as 0.1, 0.2, 0.3, etc. which means 10%, 20%, 30% of the output dropping off.

Creating a network with Dropouts

```
model = Sequential([
        Conv2D(16, (3,3), padding='same', activation='relu',
        input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(2,2),
    Dropout(0.2),
    Conv2D(32, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), padding='same', activation='relu'),
    MaxPooling2D(2,2),
    Dropout(0.2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

Testing

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

uploaded = files.upload()

for fn in uploaded.keys():
  path = '/content/' + fn
  img = image.load_img(path, target_size=(150, 150))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)

  images = np.vstack([x])
  classes = model.predict(images, batch_size=10)
  print(classes[0])
  if classes[0]>0.5:
    print(fn + " is a dog")
  else:
    print(fn + " is a cat")
```

UDACITY

Programs Career For Enterprise Sign In GET STARTED

Home > Catalog > Intro to TensorFlow for Deep Learning

FREE COURSE

Intro to TensorFlow for Deep Learning by TensorFlow

This course is a practical approach to deep learning for software developers

START FREE COURSE

NANODEGREE PROGRAM

Machine Learning Engineer

Become a Machine Learning Engineer

Accelerate your career with the credential that fast-tracks you to job success.

About this Course

COURSE COST TIMELINE SKILL LEVEL

coursera Explore What do you want to learn? For Enterprise Log In Join for Free

Browse > Computer Science > Software Development

This course is part of the **TensorFlow in Practice Specialization**

Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

★★★★★ 4.7 4,508 ratings • 899 reviews

Offered By

deeplearning.ai

Enroll for Free
Starts Nov 29

Financial aid available

100,309 already enrolled!

About Syllabus Reviews Instructors Enrollment Options FAQ

coursera Explore What do you want to learn? For Enterprise Log In Join for Free

Browse > Data Science > Machine Learning

TensorFlow in Practice Specialization

Offered By

deeplearning.ai

Enroll for Free
Starts Nov 29

Financial aid available

21,914 already enrolled!

About How It Works Courses Instructors Enrollment Options FAQ

About this Specialization

94,577 recent views

Discover the tools software developers use to build scalable AI-powered algorithms in TensorFlow, a popular open-source machine learning framework.

100% online courses

Start instantly and learn at your own schedule