A FIRE INSIDE

FINTECH PROJECT 3

Ash Ranu
Zach Steindam
Jeneia Mullins

OUR TEAM

Ash Ranu

Digital Innovation Lead - FIG

Ash team is responsible driving digital innovation for credit analysis and portfolio management for the Private Equity, Hedge Fund and Real Money Fund portfolios.

Zach Steindam

Equity Derivatives Sales

Zach is an Equity Derivatives Sales Trader for a leading agency institutional brokerage firm. He specializes in Delta One and Equity Financing trades

Jeneia Mullins

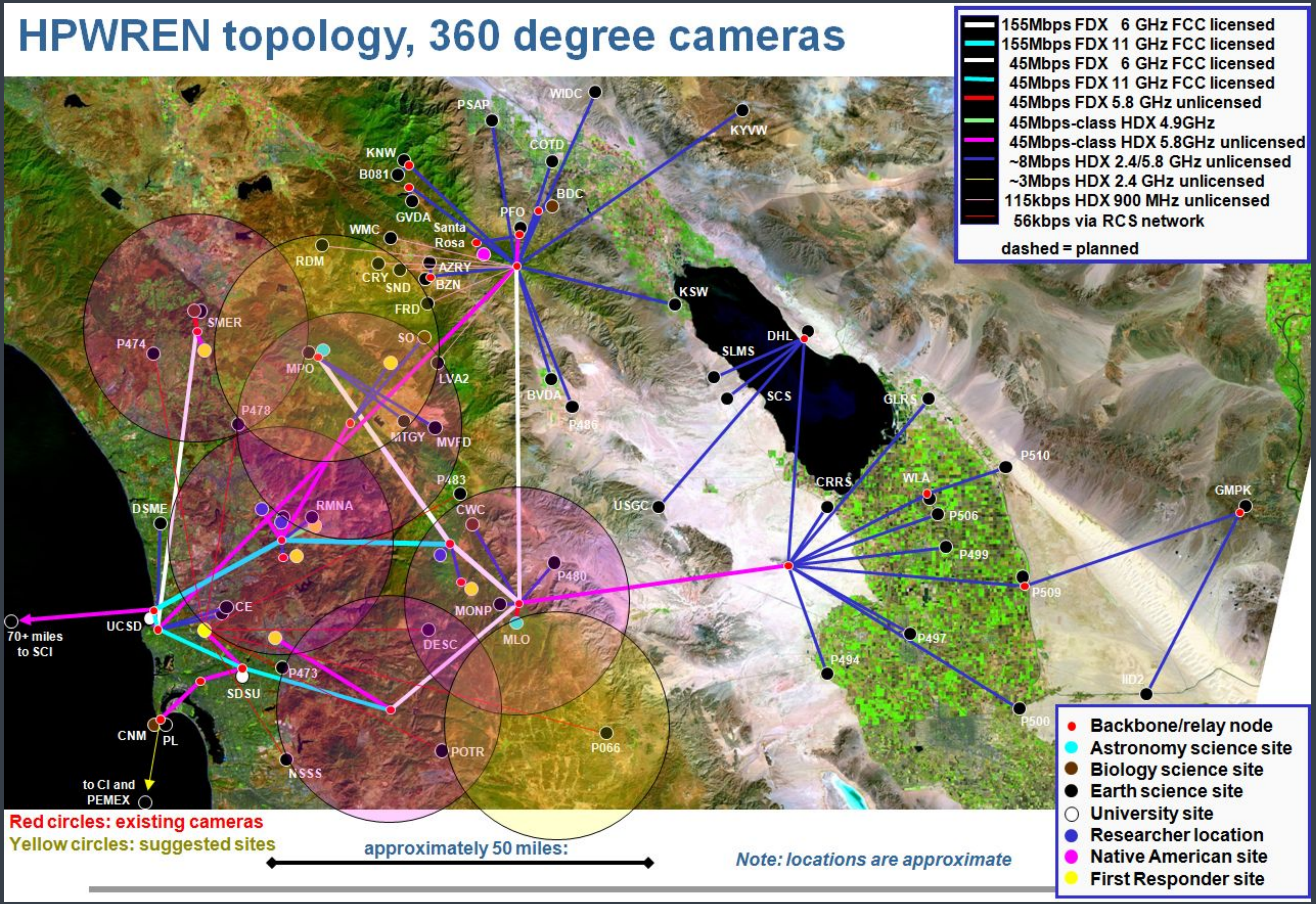
Digital Transformation

Jeneia is responsible for driving digital transformation within Middle Market Banking and Specialized Industries at JP Morgan Chase

Let’s Stop Wildfires Hackathon

- AI for Mankind launched a Let’s Stop Wildfires Hackathon with the goal to come up with ideas to help solve California wildfires crisis.
- Images captured from the High Performance Wireless Research and Education Network (HPWREN) Cameras was collated into labeled (smoke/no smoke) set of images.
- Task is to build a wildfire smoke classifier to predict if there is smoke given a small grid image.

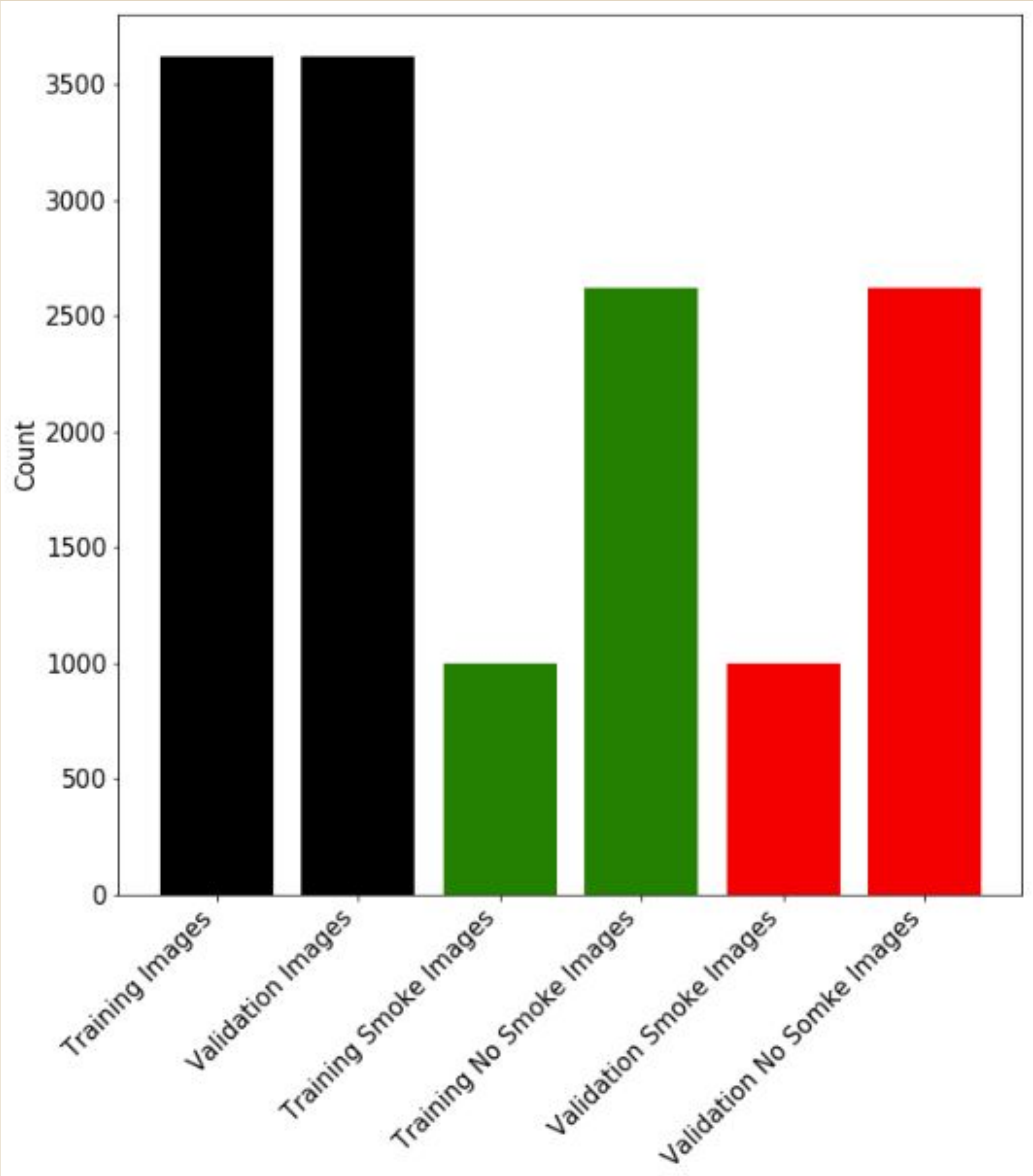
HPWREN



HPWREN topology, 360 degree cameras
155Mbps FDX 6 GHz FCC licensed
155Mbps FDX 11 GHz FCC licensed
45Mbps FDX 6 GHz FCC licensed
45Mbps FDX 11 GHz FCC licensed
45Mbps FDX 5.8 GHz unlicensed
45Mbps-class HDX 4.9GHz
45Mbps-class HDX 5.8GHz unlicensed
~8Mbps HDX 2.4/5.8 GHz unlicensed
~3Mbps HDX 2.4 GHz unlicensed
115kbps HDX 900 MHz unlicensed
56kbps via RCS network
dashed = planned
WIDC
PSAP
KYVW
COTD
KNW
B081
BDC
PFO
GVDA
WMC
Santa Rosa
RDM
AZRY
CRY
SND
BZN
KSW
FRD
SMER
DHL
P474
SO
SLMS
MPO
LVA2
BVDA
SCS
GLRS
P478
P486
MTGY
MVFD
P510
P483
CRRS
WLA
GMPK
DSME
RMNA
CWC
USGC
P506
P499
P480
P509
CE
MONP
UCSD
MLO
70+ miles to SCI
DESC
P497
P494
P473
SDSU
IID2
P500
CNM
PL
POTR
P066
NSSS
to CI and PEMEX
Red circles: existing cameras
Yellow circles: suggested sites
approximately 50 miles:
Note: locations are approximate
Backbone/relay node
Astronomy science site
Biology science site
Earth science site
University site
Researcher location
Native American site
First Responder site

DATASET

IMAGE CATEGORIES

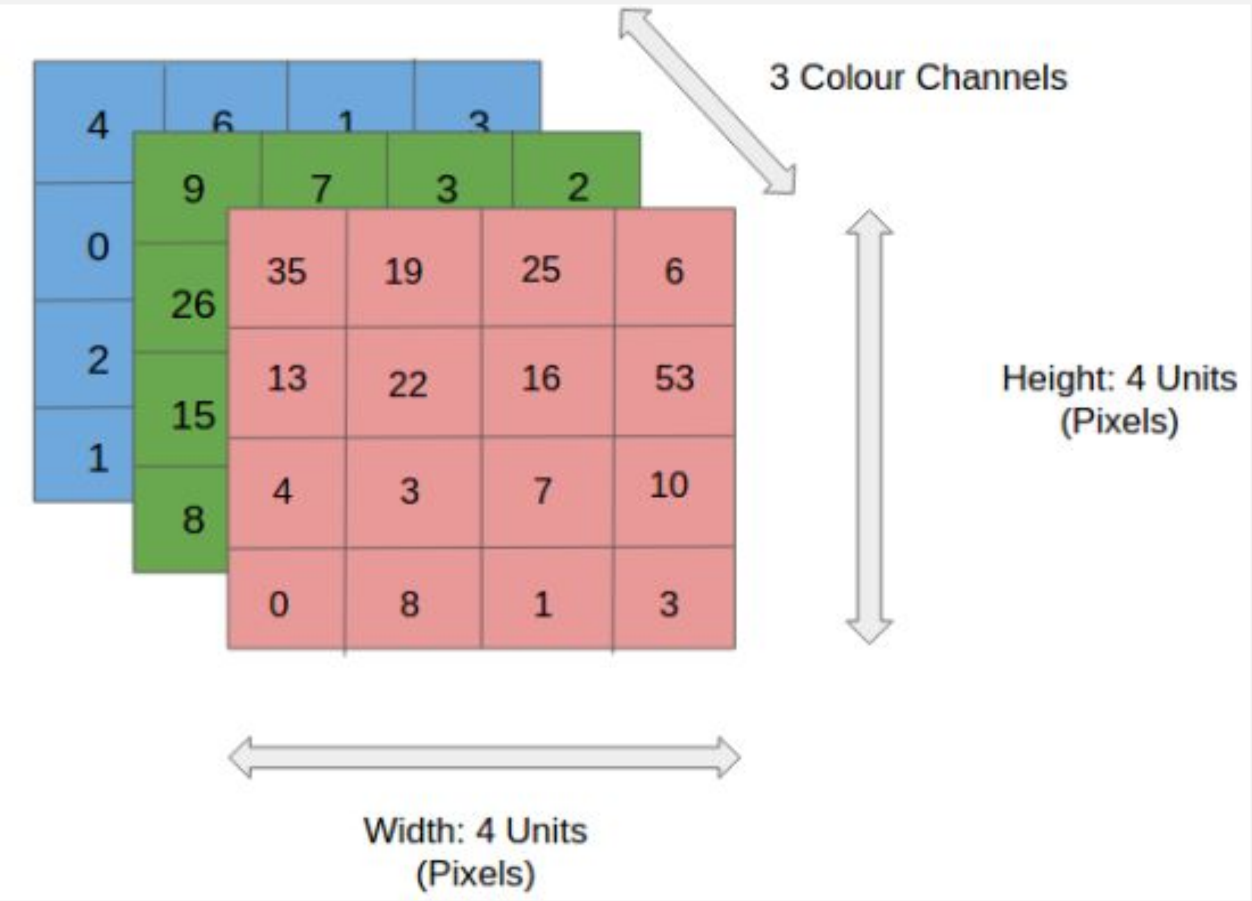


SAMPLE IMAGES

CONVOLUTIONAL NEURAL NETWORKS

INPUT IMAGE

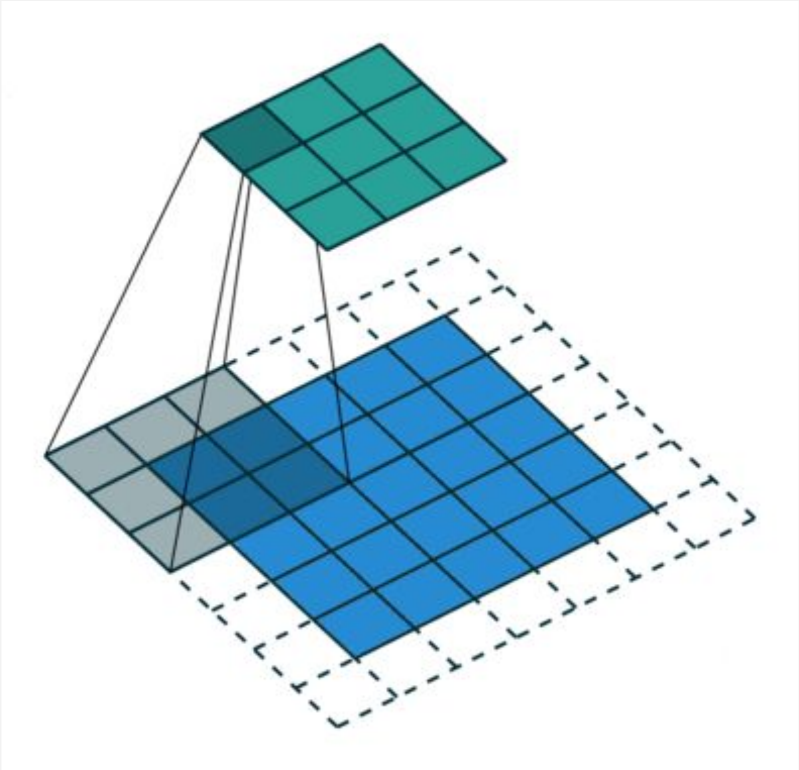
- RGB image is separated by its three color planes — Red, Green, and Blue.



INITIAL FILTER

- Convolutional networks use filters (or a documented set of features) that classify details about the image
- The network reduces the image into a form which is easier to process, without losing critical features critical for a good prediction

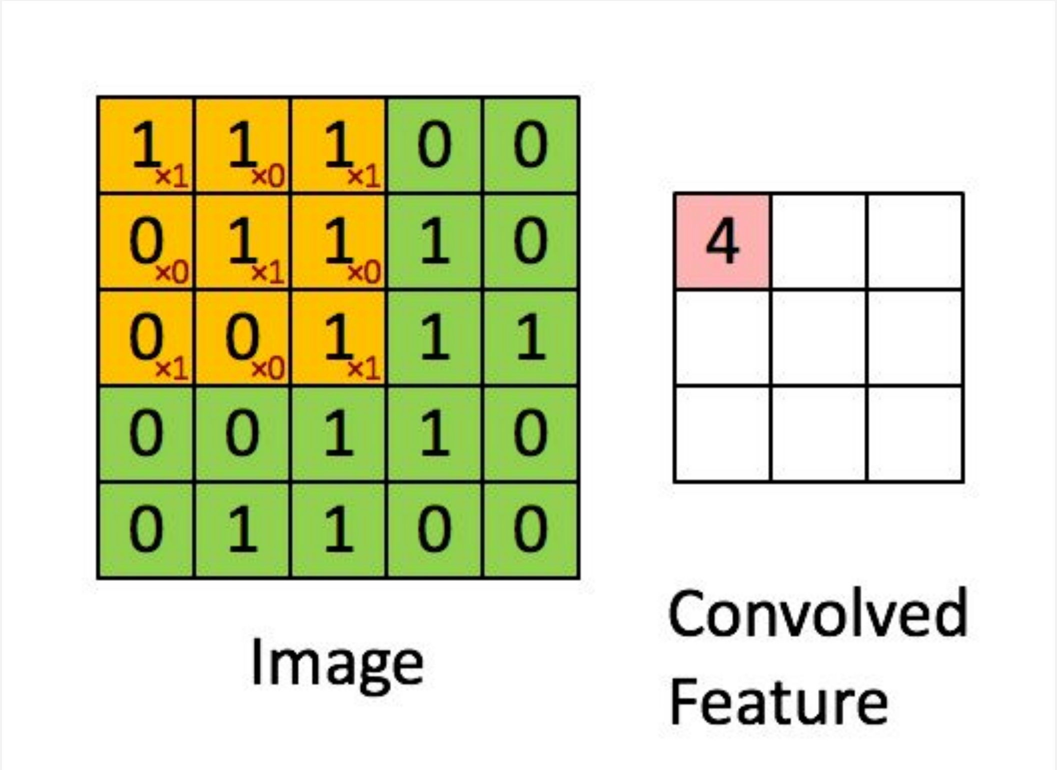
The first layer captures high-level features (such as edges, gradient, and color)



CONVOLVED FEATURES

- The filter (or kernal) will slide across the image and document a convolved feature
- The filter repeats this process until it has scanned across the entire image
- Matrix multiplication calculates the bias weights to provide a robust Convoluted Feature Output.

Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature

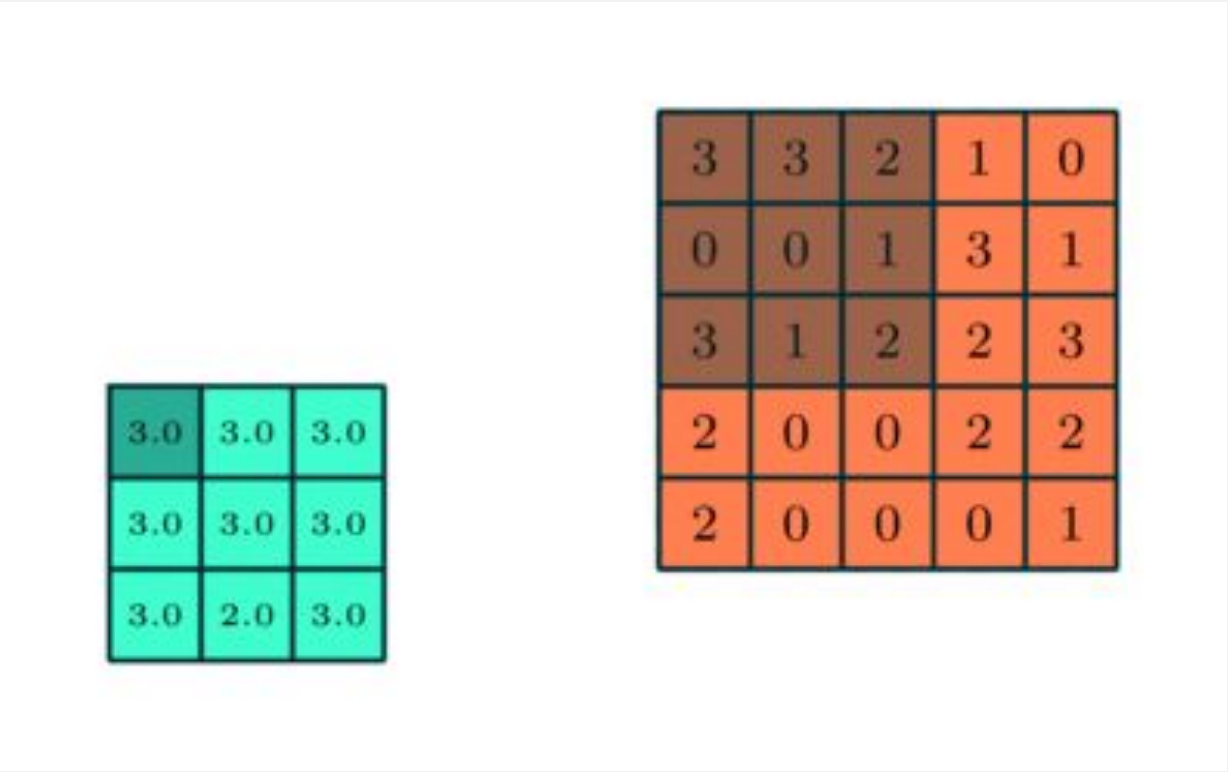


CONVOLUTIONAL NEURAL NETWORKS

MAX POOLING

- A pooling layer reduces the size of the convoluted feature in order to decrease the computational power required to process the data
- Max pooling is preferred because it acts as a noise suppressant in feature extraction
- These features are then fed into a deep neural net to measure predictions

3x3 pooling over 5x5 convolved feature



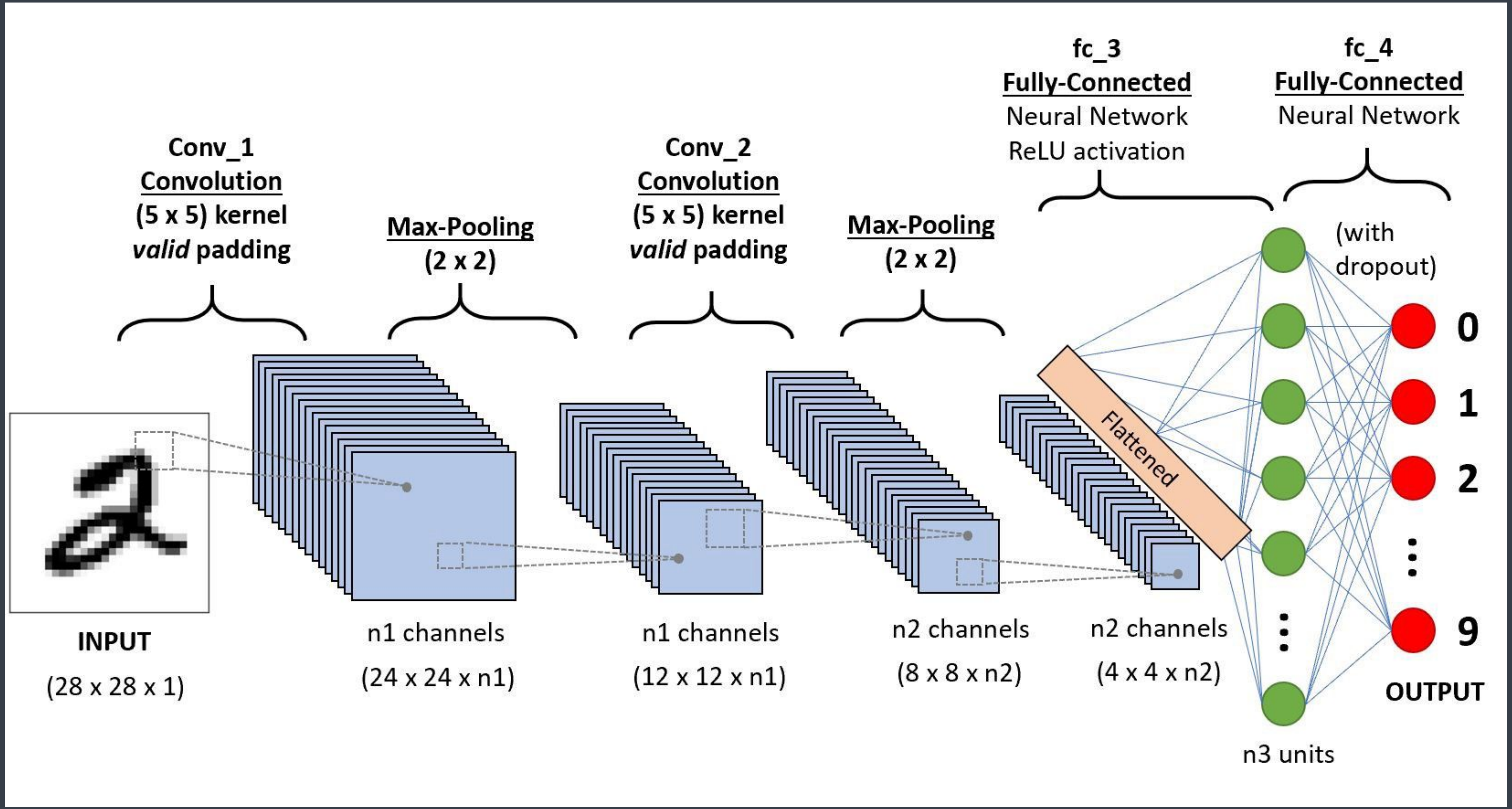
SEQUENTIAL CONV. LAYERS

- When we go through another conv. layer, the output of the first conv. layer (activation map) becomes the input of the 2nd conv. layer.

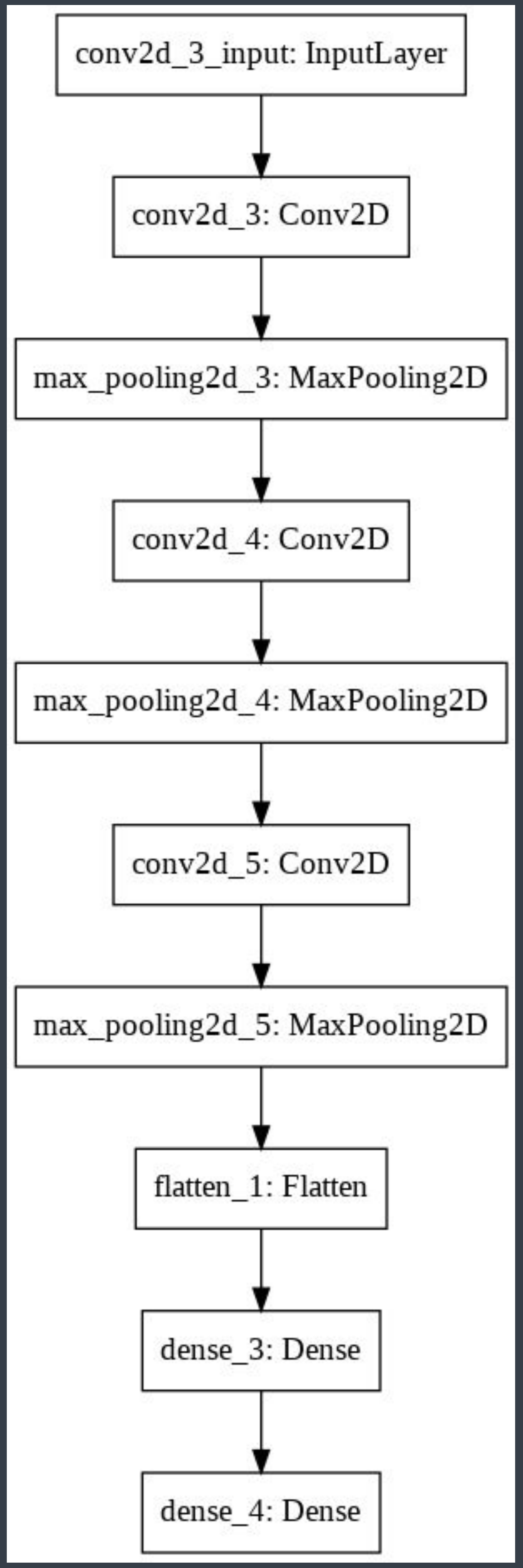
FULLY CONNECTED LAYERS

- In our custom model we use three fully connected layers as follows:
  - Fully connected input layer (flatten) —takes the output of the previous layers, “flattens” them and turns them into a single vector that can be an input for the next stage.
  - The first fully connected layer—takes the inputs from the feature analysis and applies weights to predict the correct label.
  - Fully connected output layer—gives the final probabilities for each label.

CONVOLUTIONAL NEURAL NETWORKS

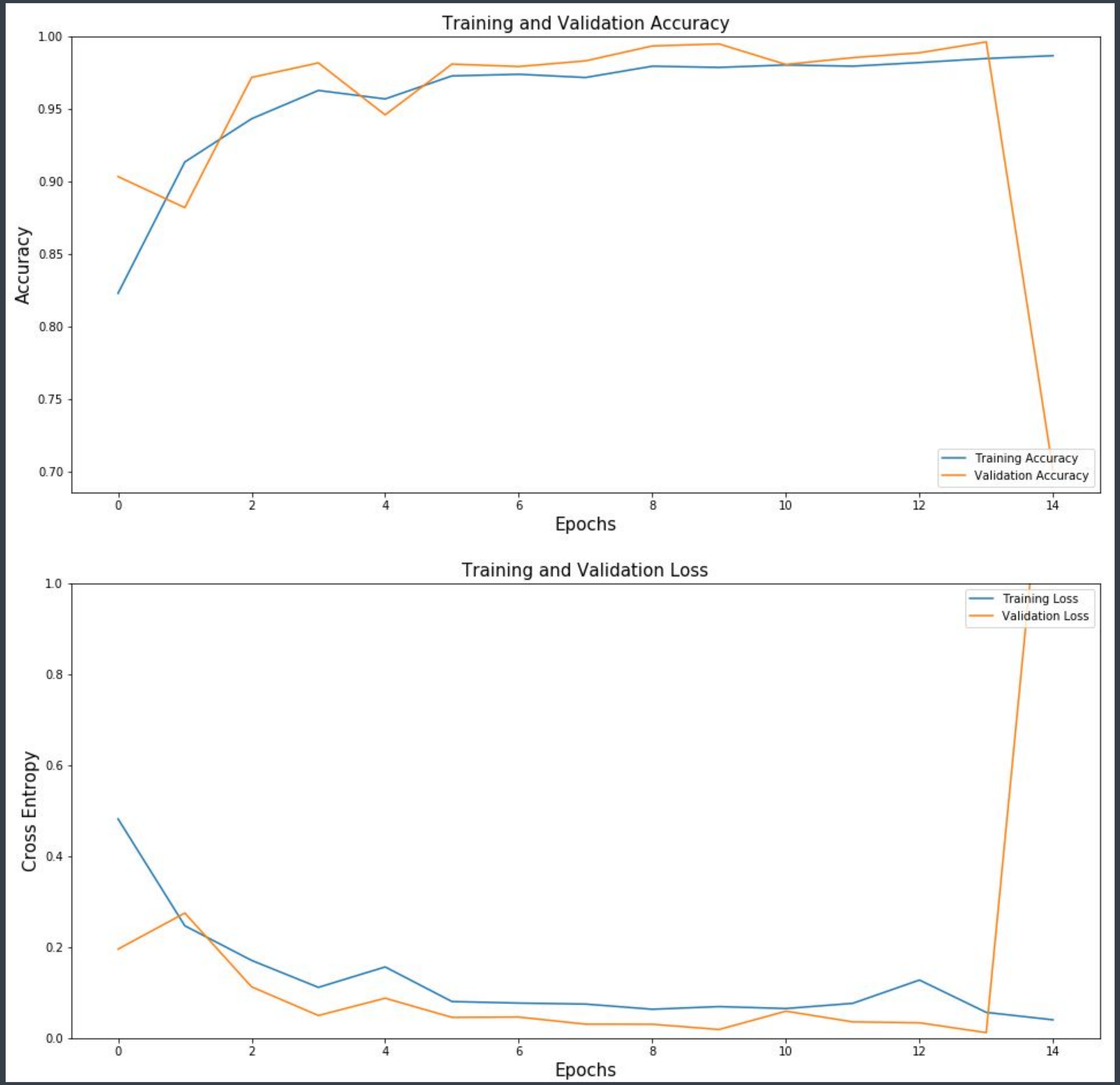


CUSTOM MODEL - 3 COV LAYERS



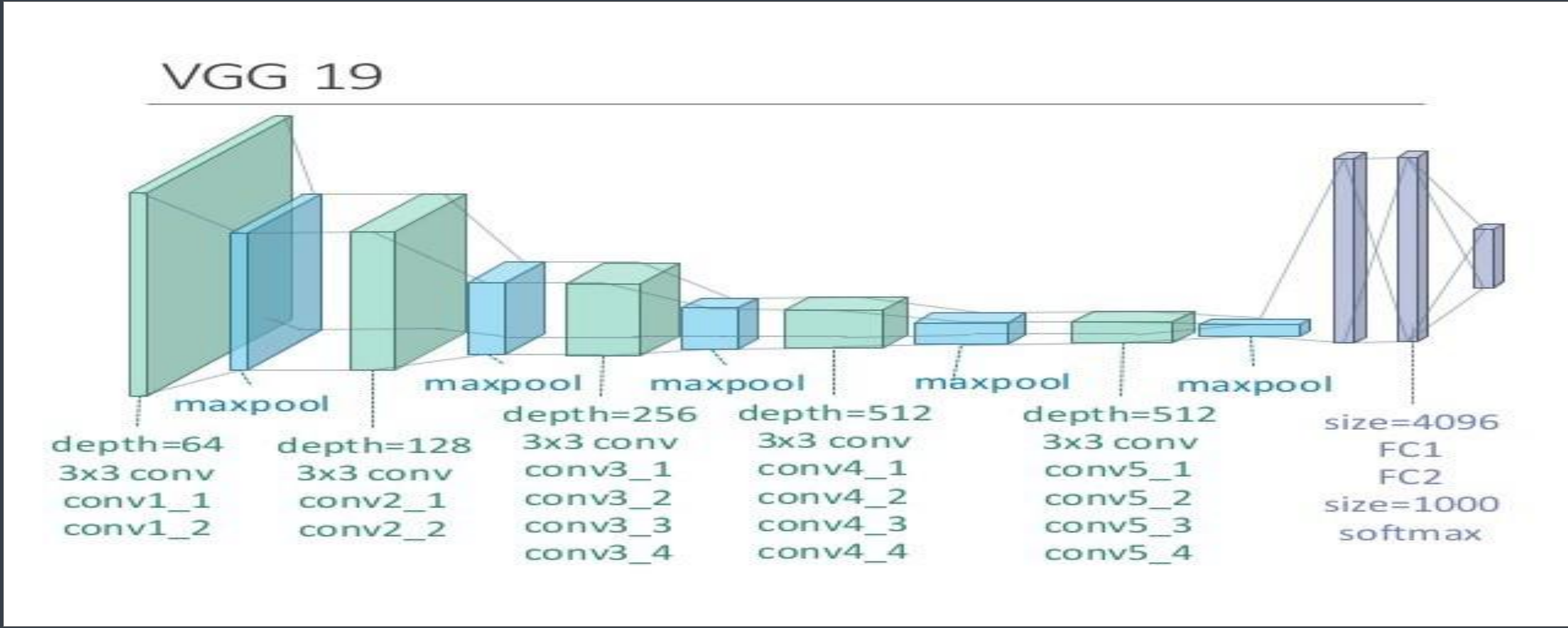
PARAMETERS

- Image Height = 224
- Image Width= 224
- Class_mode= ' binary' - Smoke or No Smoke
- loss= 'binary_crossentropy', - Return high values for bad predictions and low values for good predictions.For a binary classification the typical loss function is the binary cross-entropy / log loss.
- Dense(512, activation='relu') - Applies the rectified linear unit activation function.. Mathematically, it is defined as y = max(0, x). Visually, it looks like the following:
- Dense(1, activation='sigmoid') - Sigmoid activation function. Sigmoid is equivalent to a 2-element Softmax, where the second element is assumed to be zero.



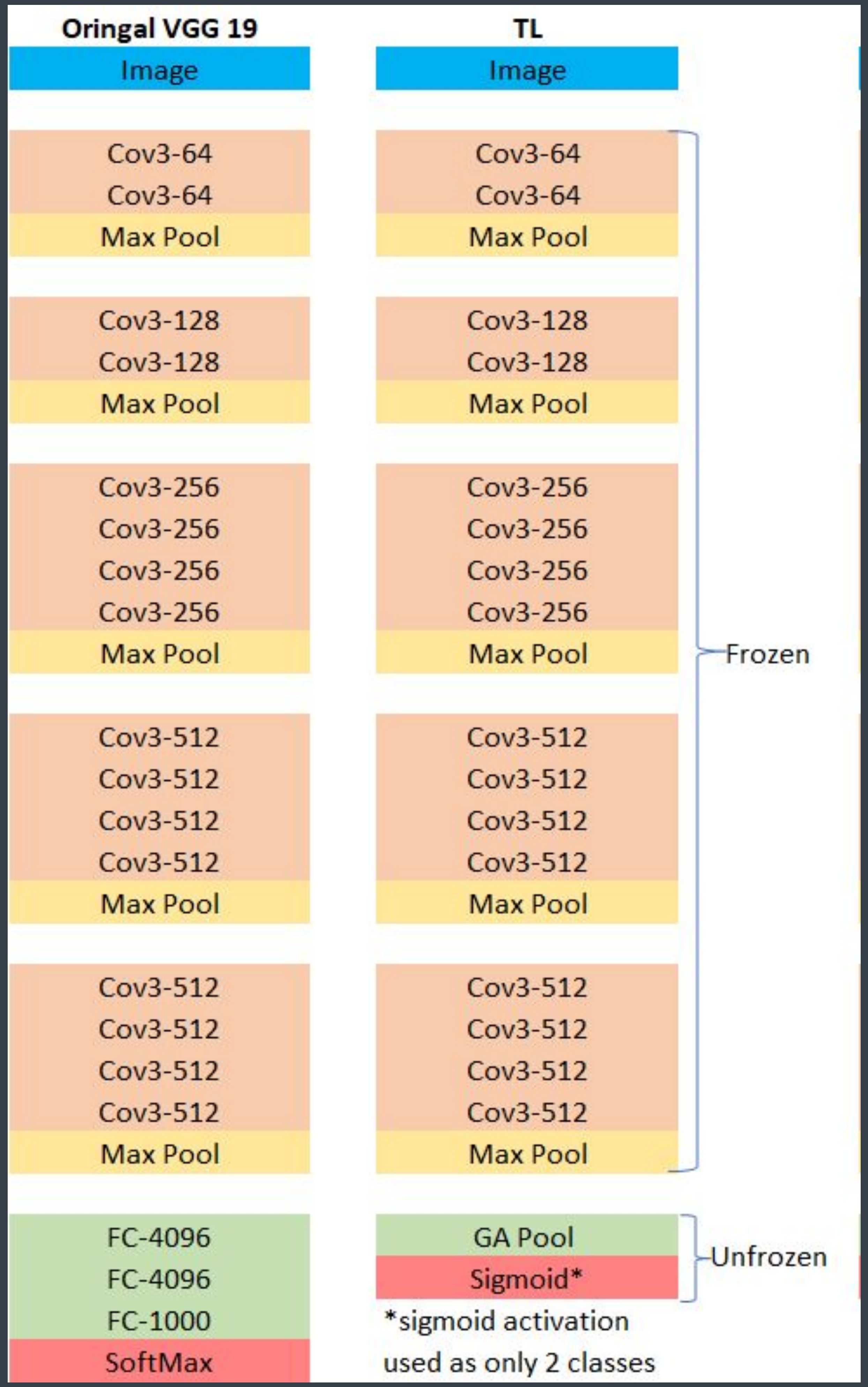
TRANSFER LEARNING VGG19

- Transfer learning uses pre-trained models and applies it to a new dataset. Data scientists often utilize transfer learning when attempting to train large datasets that can take hours and significant GPU to process on a personal computer.
- We chose VGG-19 to apply to our dataset of images. VGG-19 is a convolutional neural network that is trained on millions of images from the ImageNet database. VGG uses smaller 3x3 filters as they have the same effective field as one 7x7 filter.
- The network is 19 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.



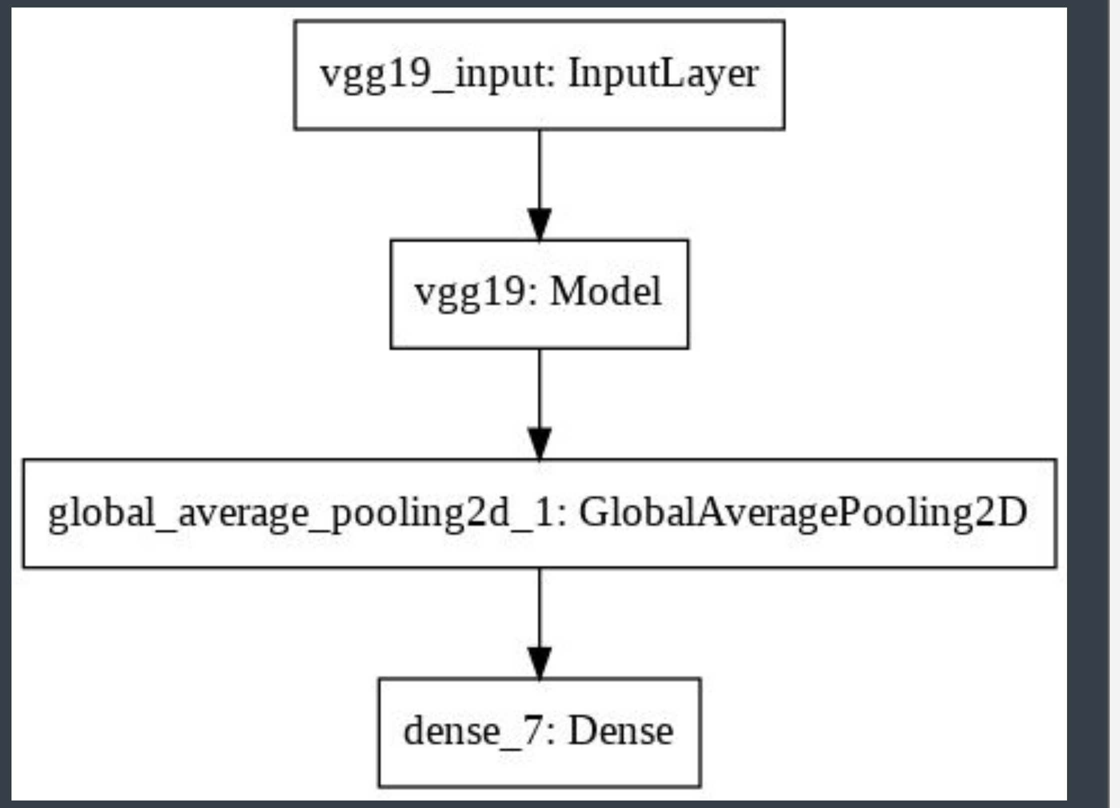
TRANSFER LEARNING VGG19

```
Number of layers in the base model:  22
<tensorflow.python.keras.engine.training.Model object at 0x7f50f55c6ef0>: False
<tensorflow.python.keras.layers.pooling.GlobalAveragePooling2D object at 0x7f50f3cae160>: True
<tensorflow.python.keras.layers.core.Dense object at 0x7f50f3caef60>: True
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Model)                (None, 7, 7, 512)         20024384
_________________________________________________________________
global_average_pooling2d_3 ( (None, 512)               0
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 513
=================================================================
Total params: 20,024,897
Trainable params: 513
Non-trainable params: 20,024,384
_________________________________________________________________
```



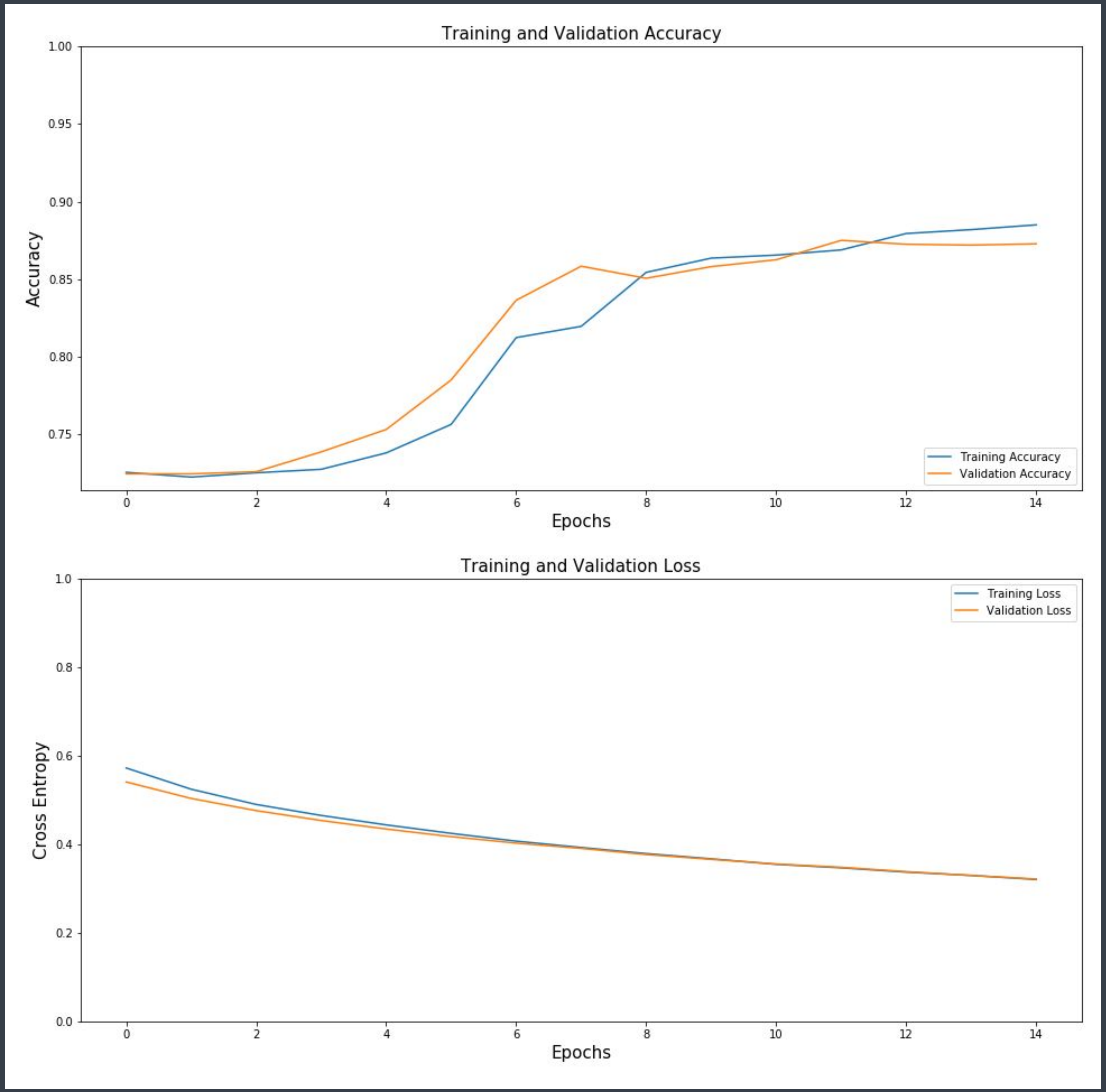
Left: The original VGG network architecture.
Right: Removing the original FC Layers and replacing them with a brand new FC head.

TRANSFER LEARNING VGG19



PARAMETERS

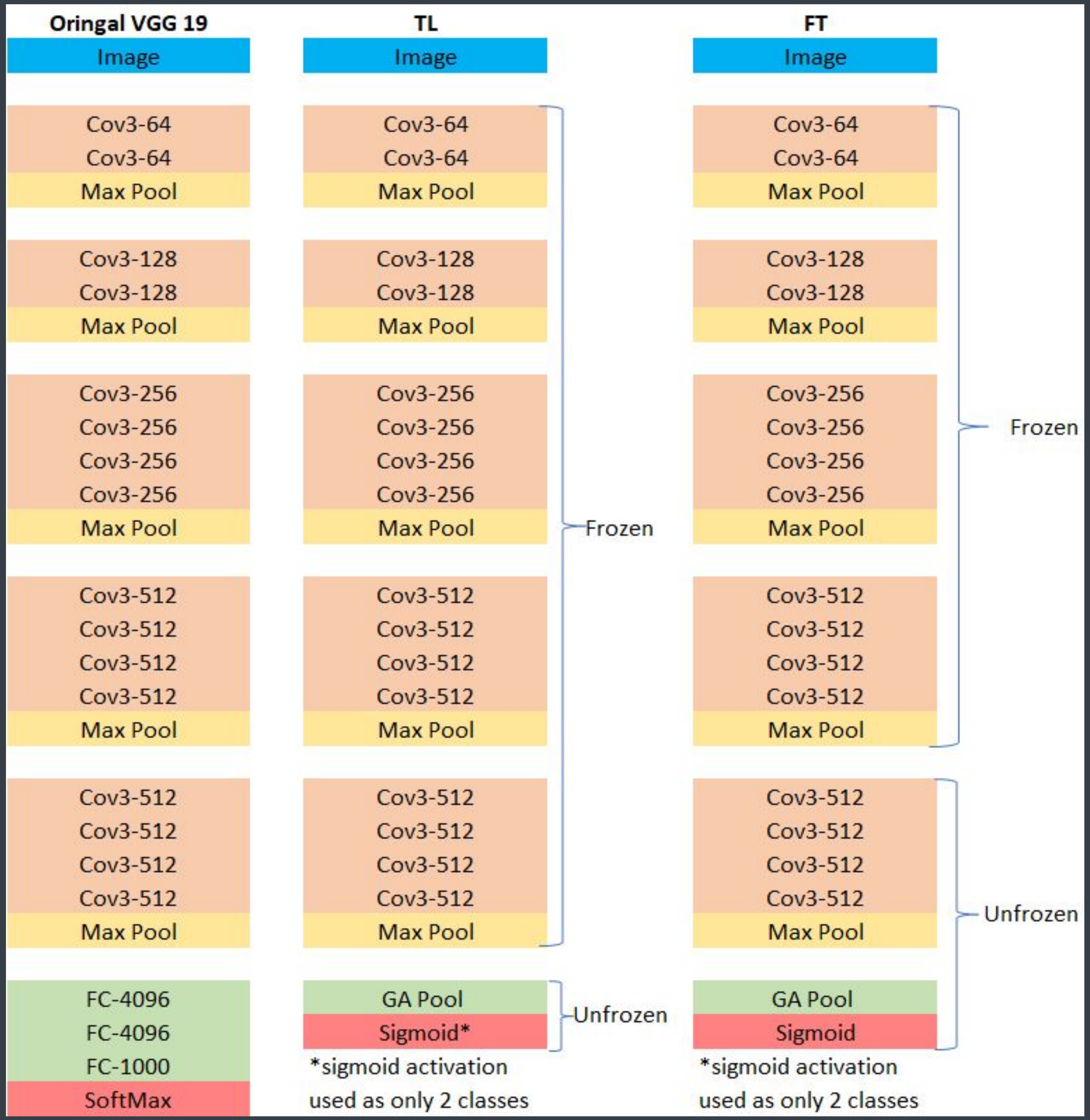
- include_top=False,weights='imagenet' - Create a base model using VGG19 with imagenet weights and without the predictive layer
- Adding in Fully Connected Layers
- base_model.trainable = False - Only FC layers will be trained.
- batch_size=30
- class_mode = 'binary'
- total_classes = 1
- activation_function = 'sigmoid'
- loss_function = 'binary_crossentropy'base_model.trainable = False
- reload_checkpoint=True



FINE TUNING

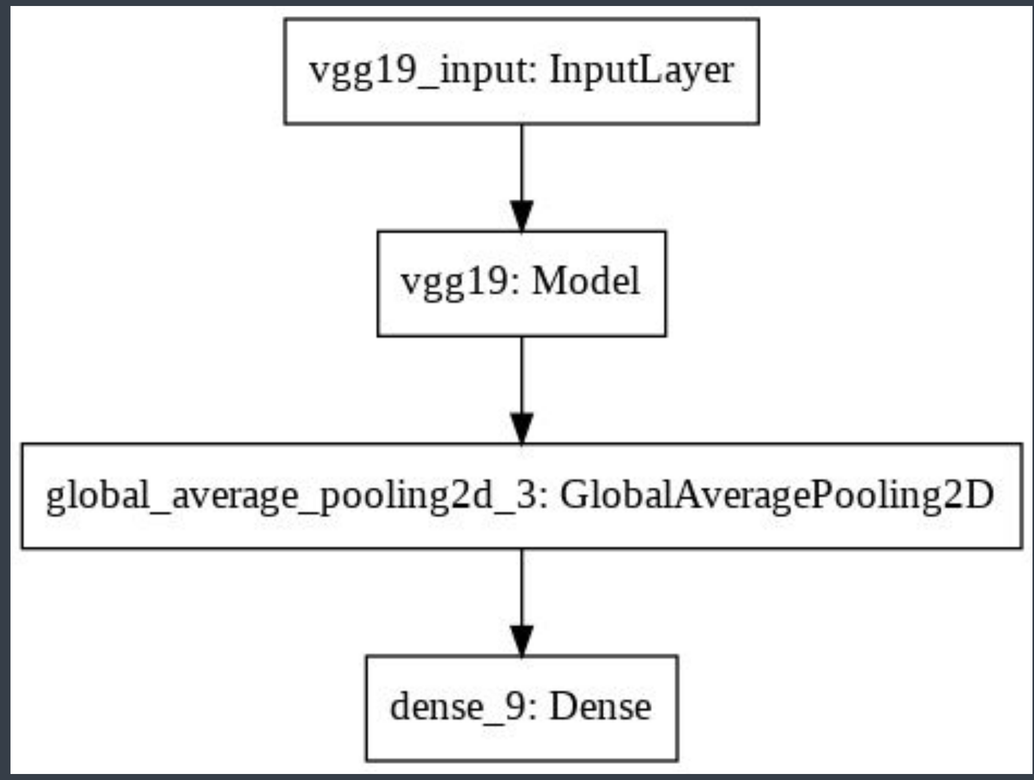
Fine Tuning

```
#Fine-tune a subset of top layers

#Unlock base model layers
base_model.trainable = True

#reset data generators
train_generator.reset()
validation_generator.reset()

# Fine tune start from layer 17
fine_tune_at = 17

# Freeze layers before fine tuning layer
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False

#lower the learning rate
model.compile(optimizer=Adam(lr=0.0001),
              loss='binary_crossentropy',
              metrics=['accuracy'])


print("Number of layers in the base model: ", len(base_model.layers))

# loop over the layers in the model and show which ones are trainableor not
for layer in base_model.layers:
  print("{}: {}".format(layer, layer.trainable))

model.summary()
```



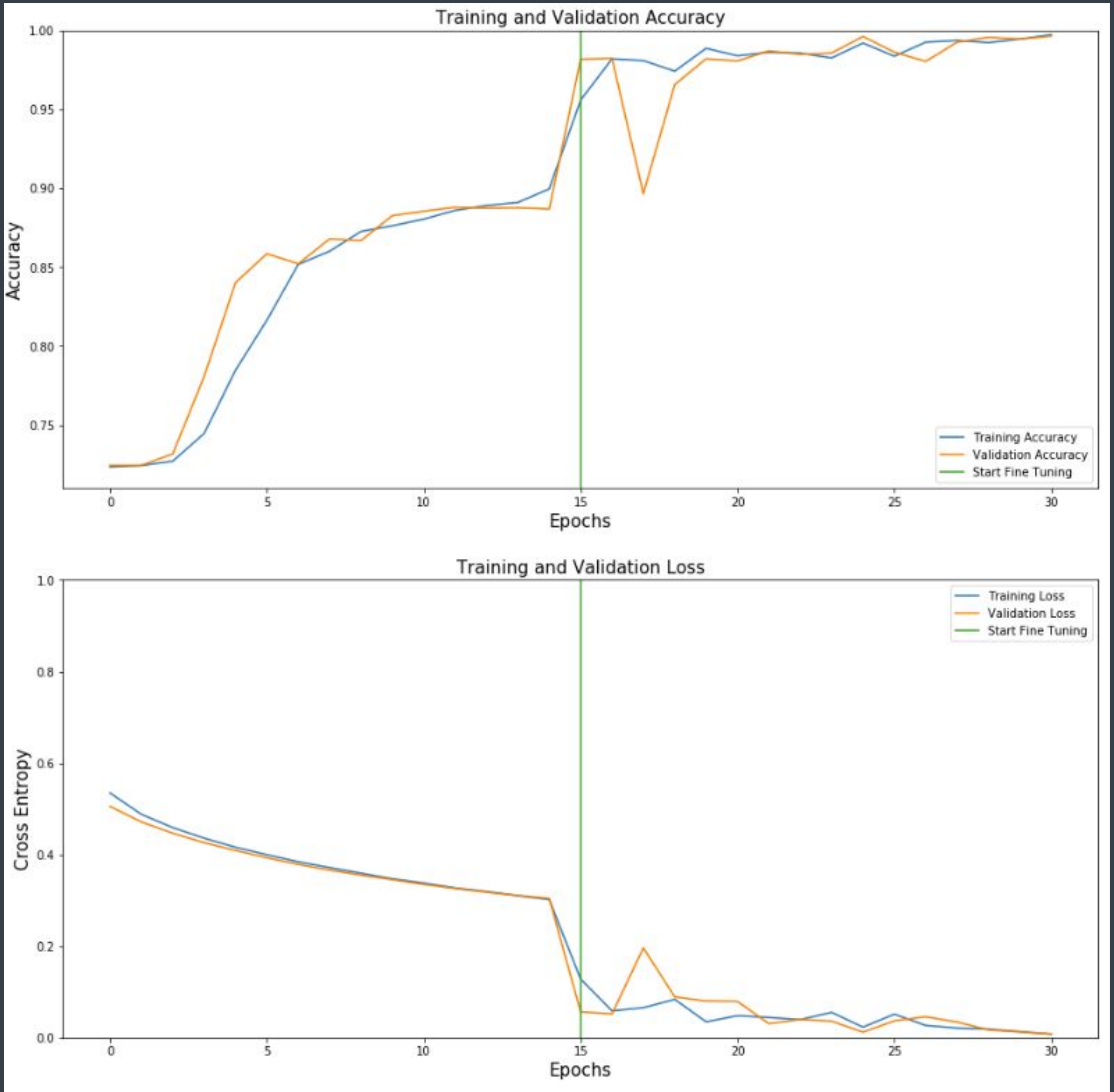
Left: The original VGG network architecture.
Middle: Removing the original FC Layers and replacing them with a brand new FC head.
Right: Fine Tuning with 4 Cov Layers and 1 Max Pooling Layer Unfrozen

FINE TUNING

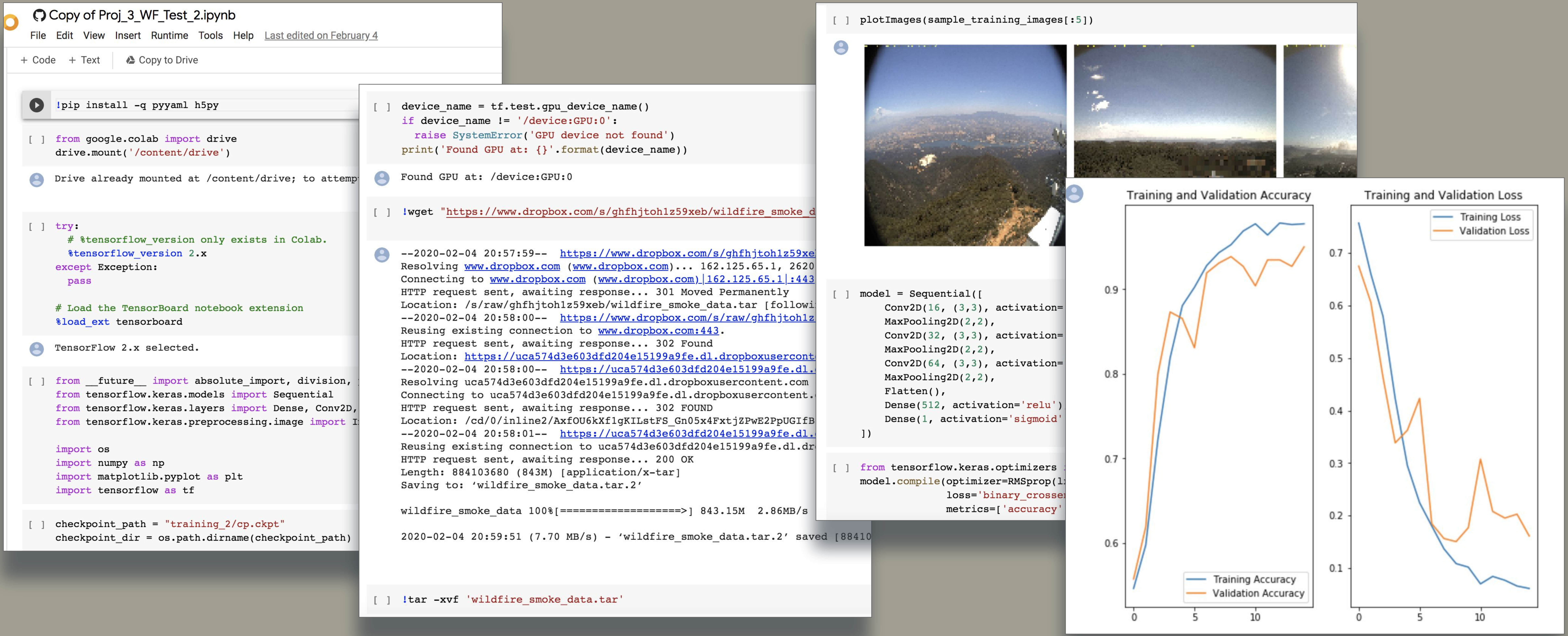


PARAMETERS

- base_model.trainable = True - Unfreeze the VGG19 lower layers and make them trainable.
- fine_tune_at = 17 - Fine tune start from layer 17 (5 unfrozen layers)
- train_generator.reset() - reset our data generators
- validation_generator.reset() - reset our data generators
- lr=0.0001 - Lower the learning rate for Fine Tuning.



BUILDING CNN MODEL IN COLAB

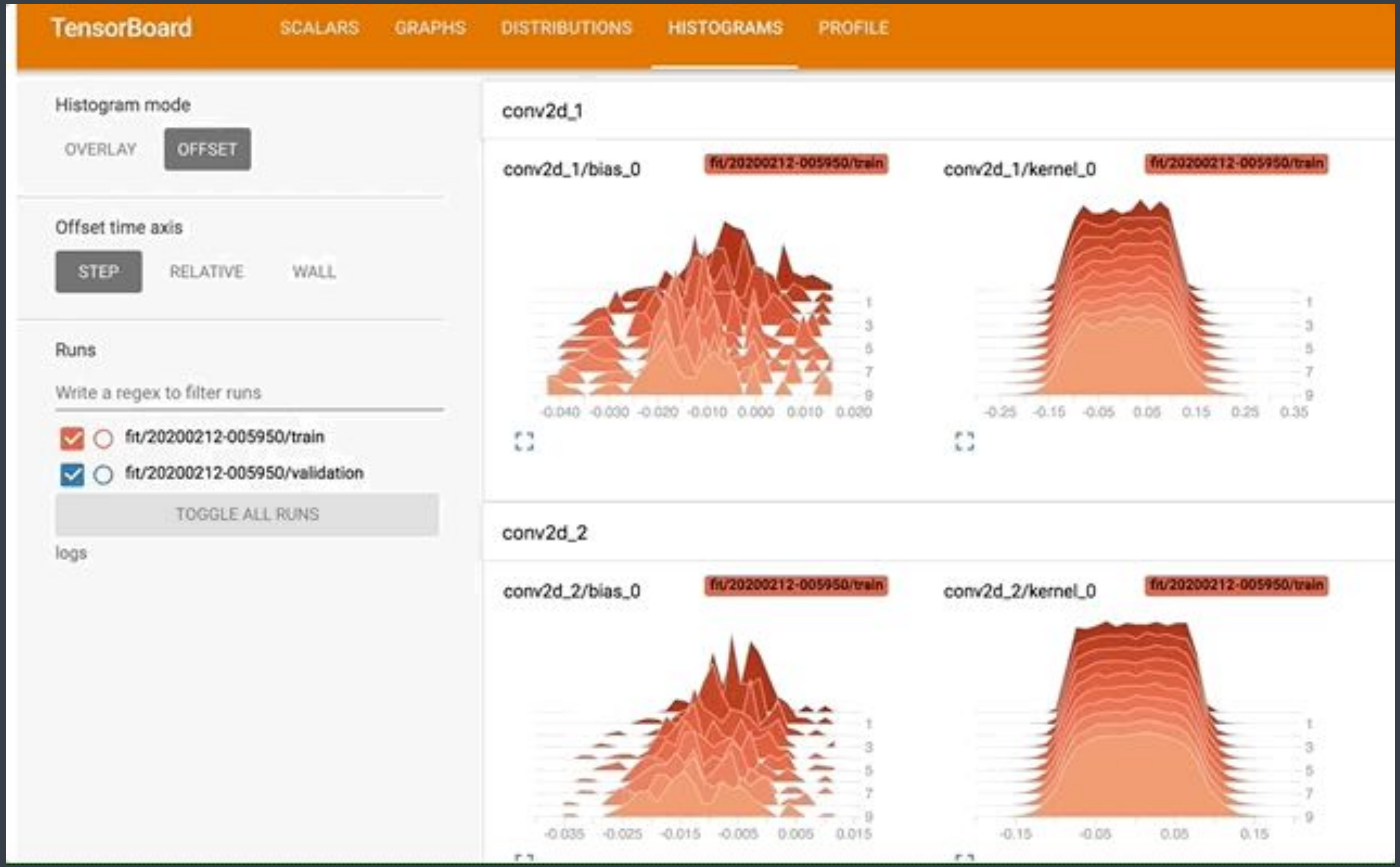


Colab allows us to use a cloud compute resource (including GPU) for free and sync our notebooks with GitHub. It was a huge time saver.

TENSORBOARD HIGHLIGHTS

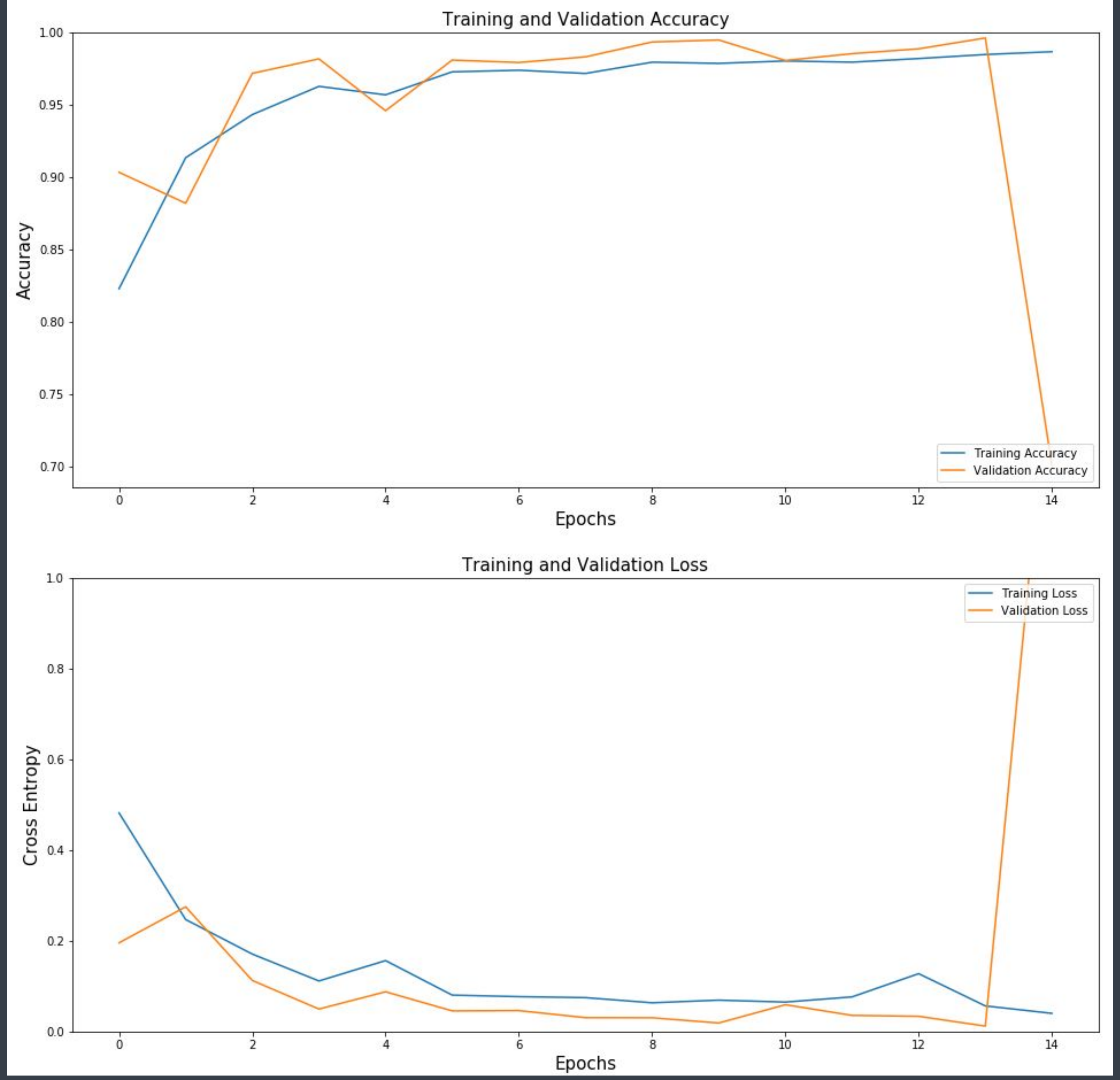
We Utilized Tensorboard Visualization Tools with Python in Jupyter and Colab

As you can see below, the accuracy rates with the TensorFlow Convolutional Neural Network returned relatively strong accuracy rates running approximately 10 epcohs

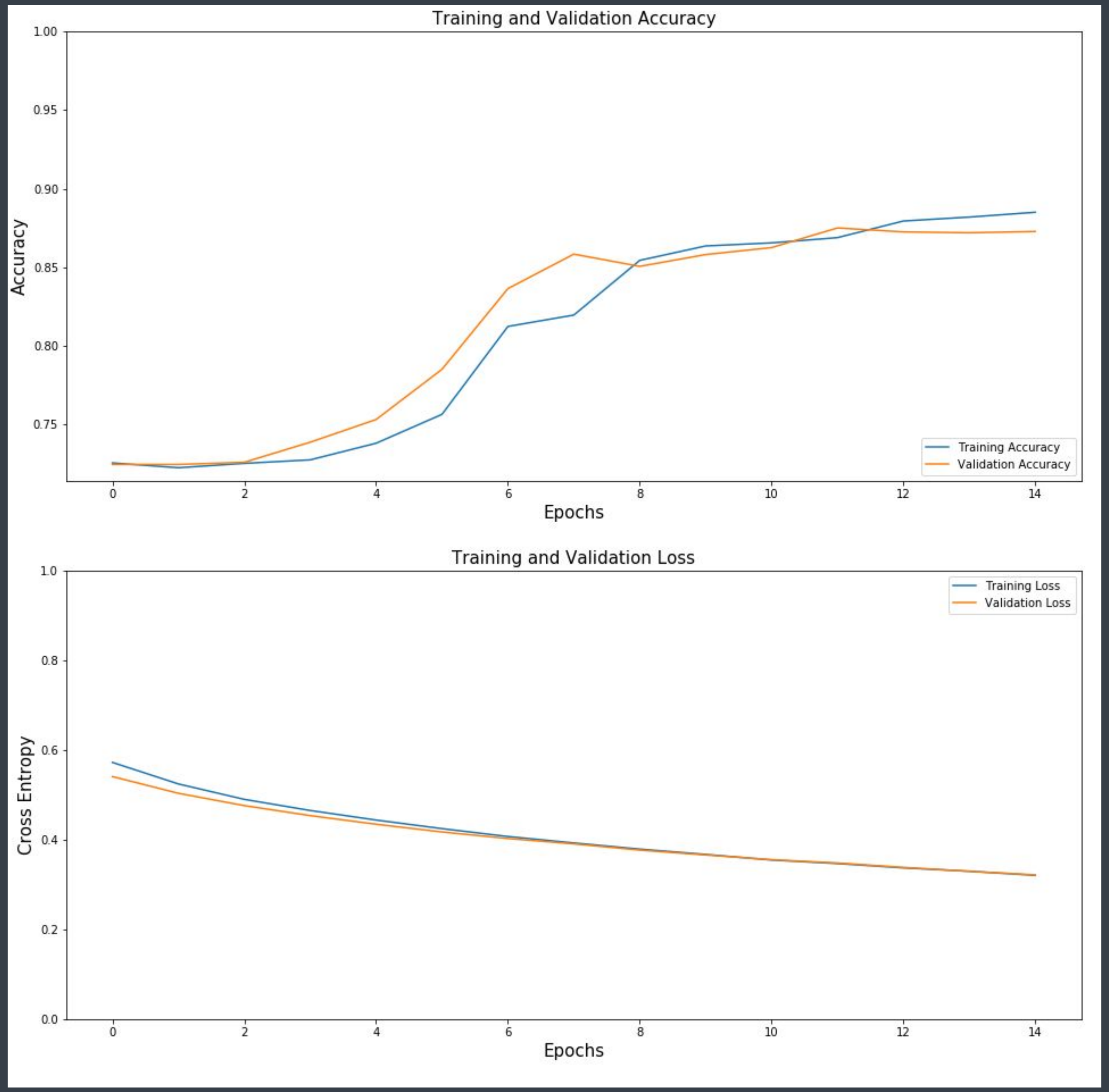


RESULTS

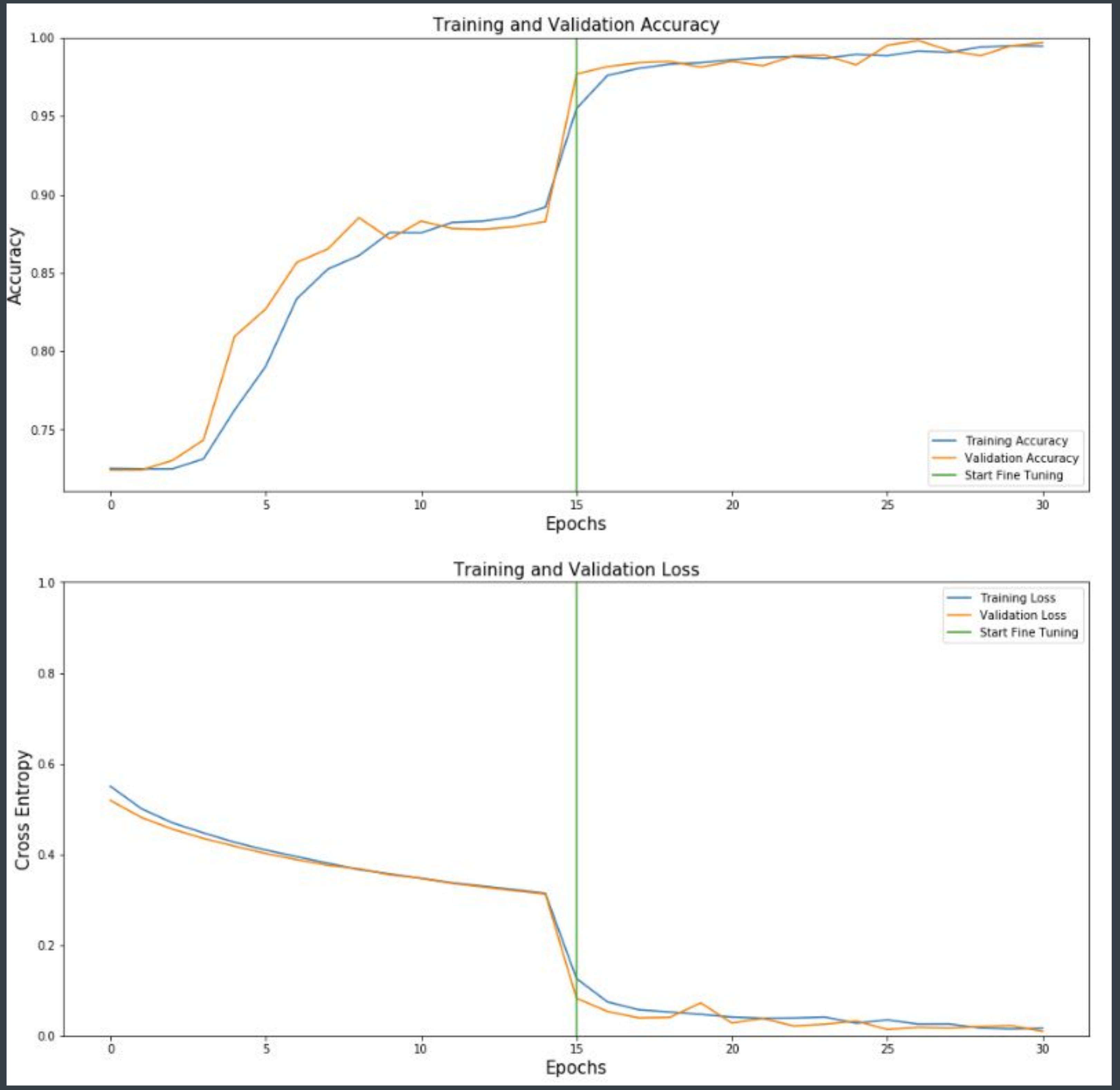
CUSTOM MODEL



TRANSFER LEARNING - VGG19



FINE TUNING MODEL



FINALIZING HACKATHON SUBMISSION

HACKATHON CHALLENGE GUIDELINES

Let's Stop Wildfires Hackathon

This hackathon is conducted by AI For Mankind. The goal is to come up with ideas to help solve
and we believe that open sharing and collaboration are important in accelerating innovation and
change locally and globally. Public and private partnerships coupled with citizen participation ca
against wildfires. Everything built during the hackathon will remain as open source with The MIT
FUEGO to provide guidances to make this hackathon possible.

Note: If you wish to keep your idea/project private, please do not enter this hackathon.

To participate in the hackathon: Register here

SUBMISSION GUIDELINES

Submission

You will have to create a new github repo for your codes with the MIT License. To allow us to evaluate y
against our test dataset, you can either

1. Save your model in HDF5 format and share it in your repo. If the model size is over 100MB, please p
Dropbox or Google Drive link to your model. We will load your model and evaluate it against our tes

or

2. Provide a python program to read in a folder consists of multiple images and predict whether there
smoke in each image and write the output in a CSV file with the following format.

Output File

image_name, smoke_detected(1 if smoke detected or 0 if no smoke)

Example:

```
image_name_a,1
image_name_b,0
```

OUR GITHUB REPO SUBMISSION!

WIVIV / **Project_3_AFI**

Wa

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Wiki | Security

No description, website, or topics provided.

12 commits | **1** branch | **0** packages | **0** releas

Branch: master | New pull request | Create new file | Uplo

zsteindam Created using Colaboratory

Copy_of_Preprocessing.ipynb	Created using Colaboratory
Copy_of_Proj_3_WF_Test_2.ipynb	Created using Colaboratory
Copy_of_get_started.ipynb	Created using Colaboratory
Copy_of_graphs.ipynb	Created using Colaboratory
Proj_3_WF_TL_2.ipynb	Created using Colaboratory
Proj_3_WF_Test_2.ipynb	Created using Colaboratory
Proj_3_WF_Test_3.ipynb	Created using Colaboratory
Proj_3_WF_Test_Transfer_Learning.ipynb	Created using Colaboratory
README.md	Initial commit

QUESTIONS?