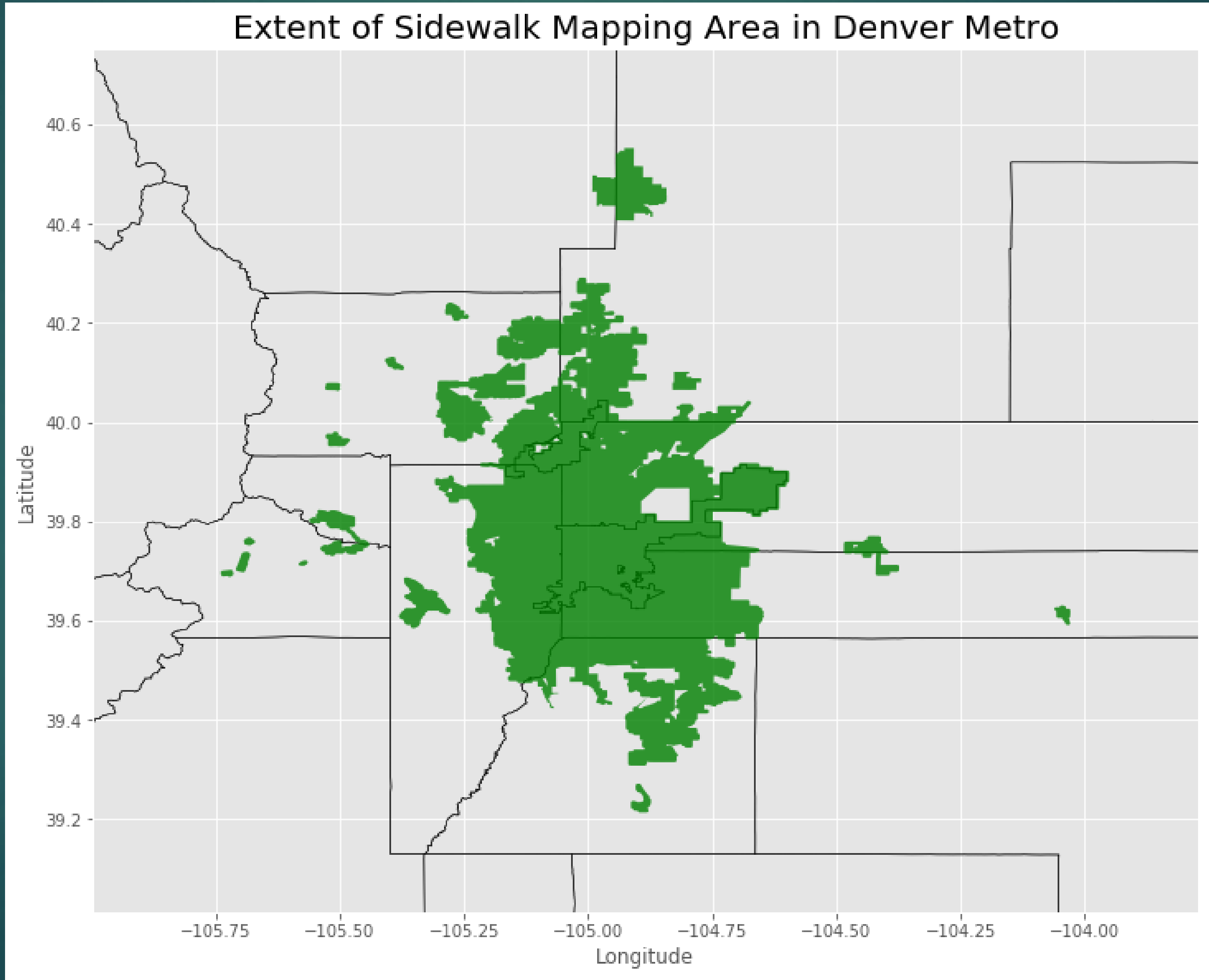
Sidewalk Image Classification and Segmentation

Data Scope:



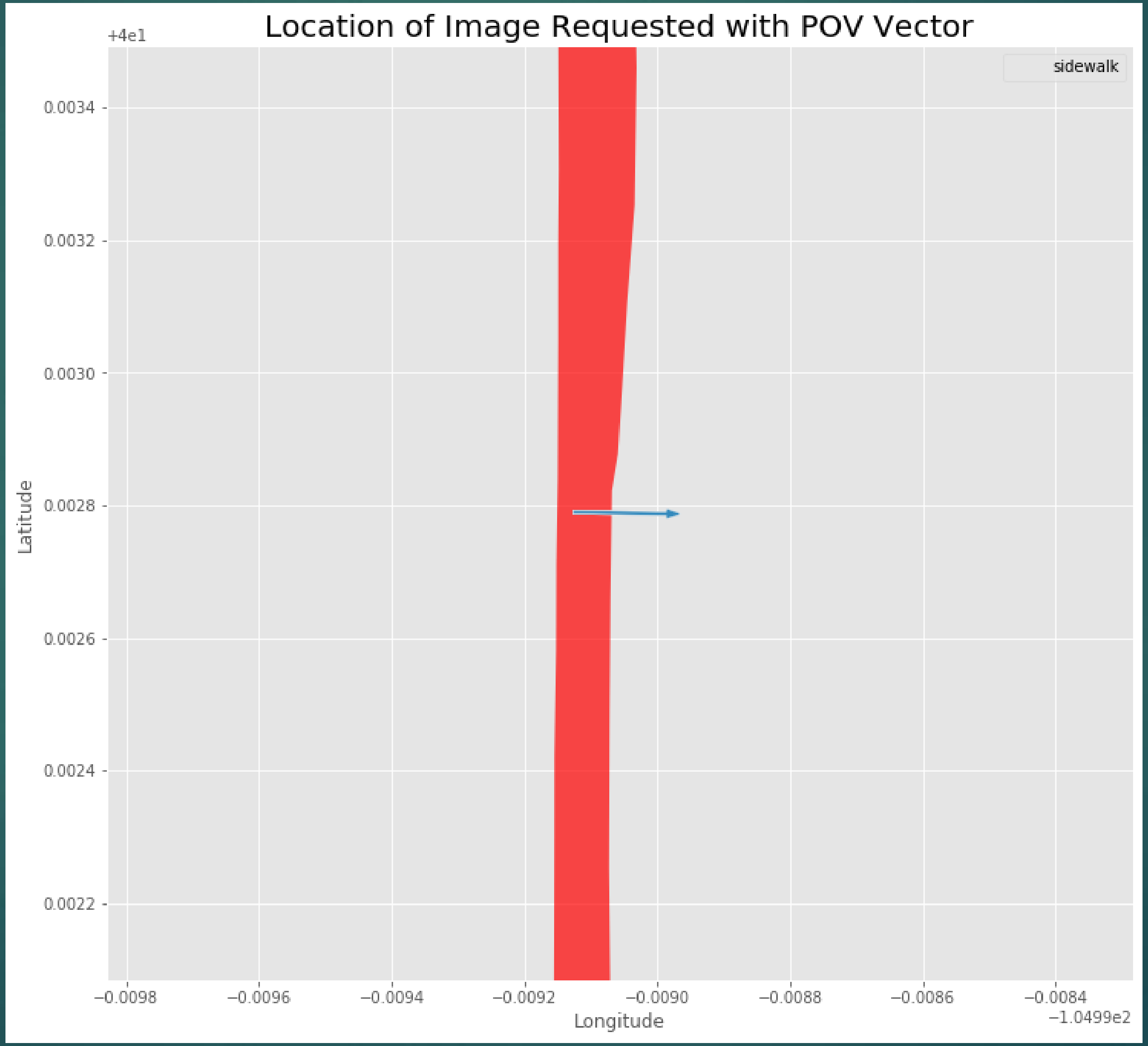
Extent of Sidewalk Mapping Area in Denver Metro
40.6
40.4
40.2
40.0
39.8
39.6
39.4
39.2
Latitude
−105.75
−105.50
−105.25
−105.00
−104.75
−104.50
−104.25
−104.00
Longitude

Image Downloading Class

- Class to download images with Latitude, Longitude, Heading, and Label
- Generate Random point within survey region
- Query streetview API for nearest image point
- Confirm point is within street polygon
- Find nearest street polygon edge & convert to direction
- Test if sidewalk exists in direction within set distance
- Save parameters and download image

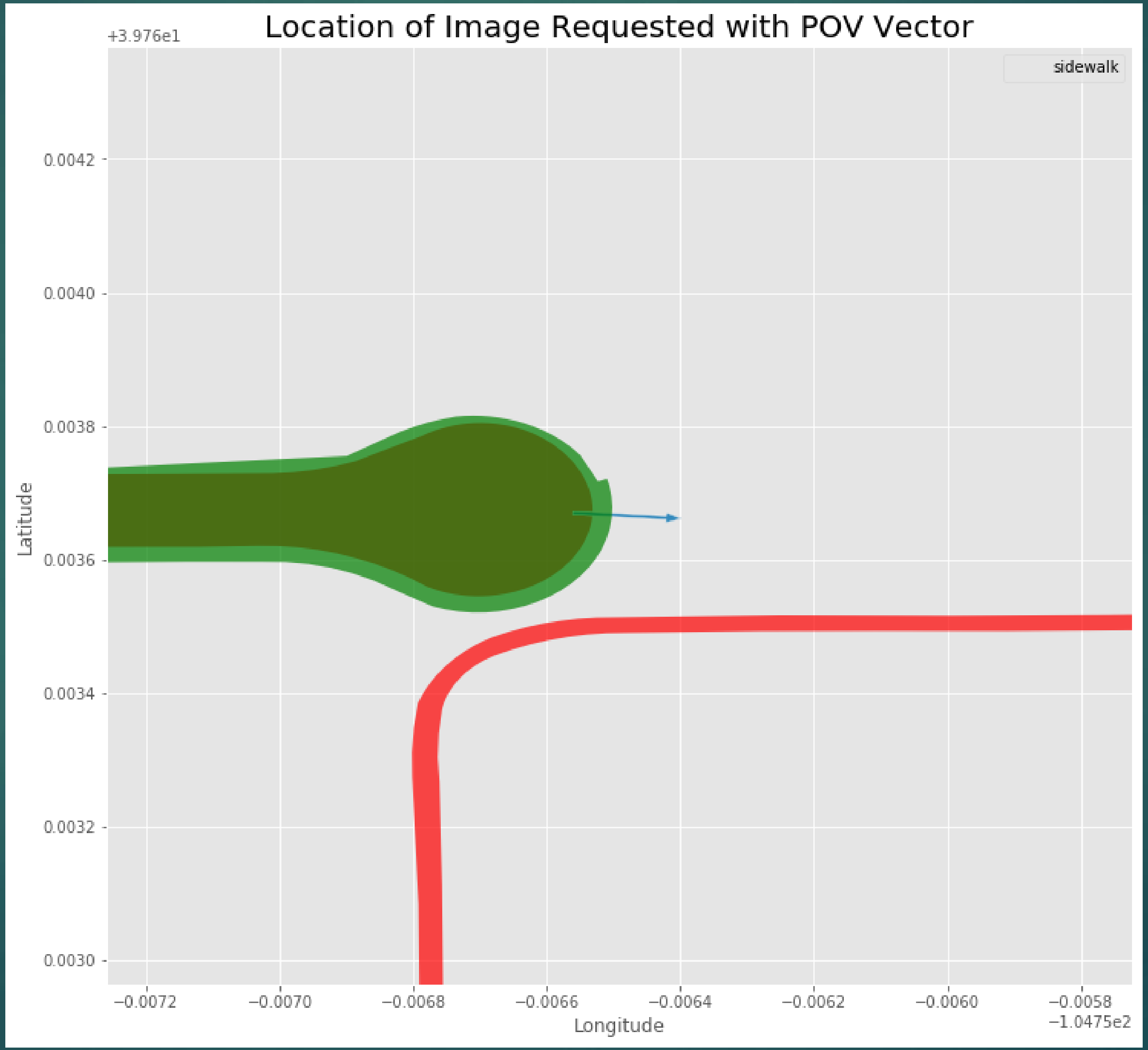
Example Images with Validation

[268.79833469817083, [40.00278986277844, -104.9991271379266], 'no_sidewalk’]



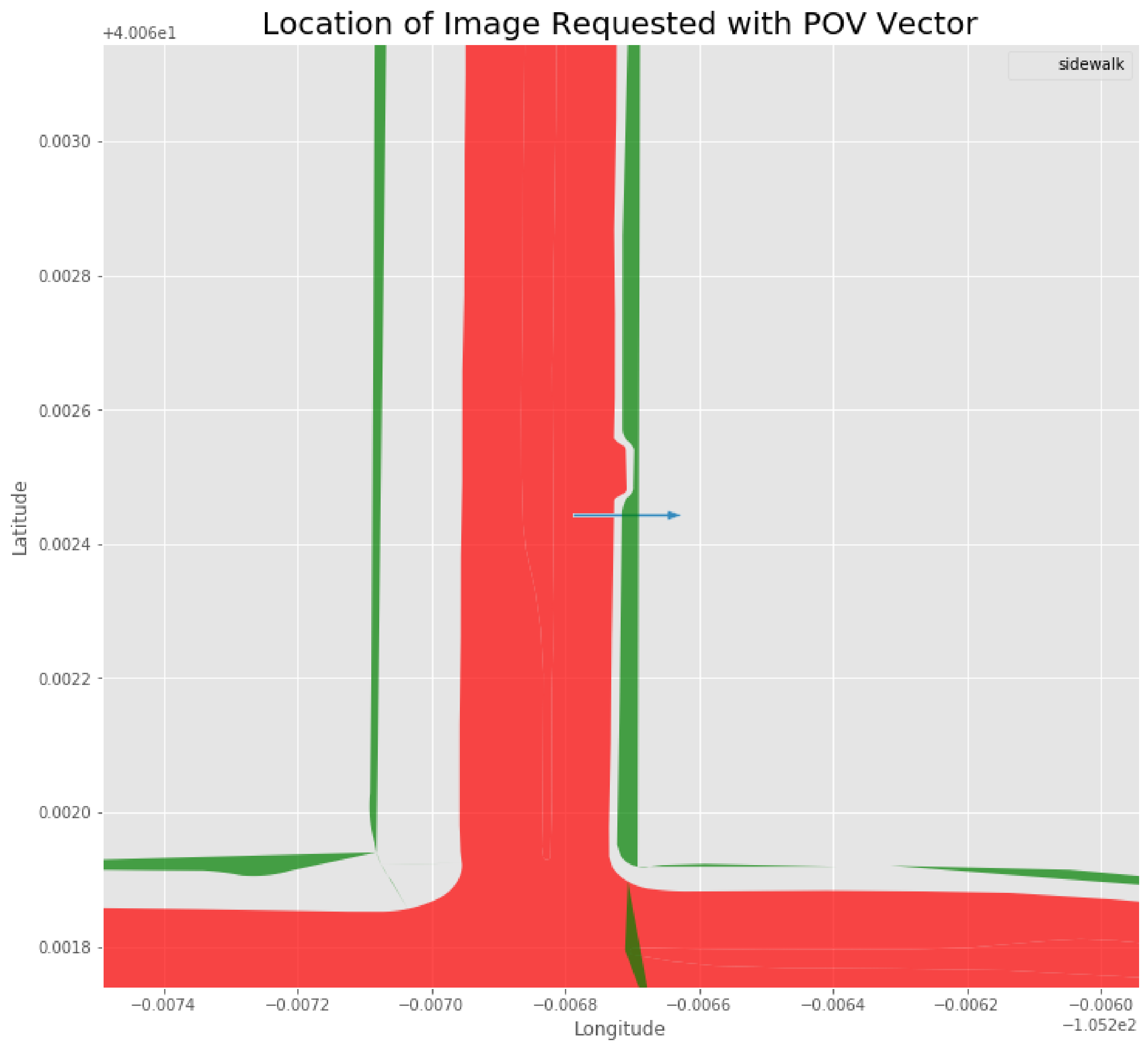
[268.79833469817083, [40.00278986277844, -104.99127137929266], 'no_sidewalk’]

[93.8101431467257, [39.7636699, -104.7565609], 'sidewalk']



[93.8101431467257, [39.7636699, -104.7565609], 'sidewalk']

[269.9812922349449, [40.06244274355016, -105.2067883010862], 'sidewalk']



[269.9812922349449, [40.06244274355016, -105.2067883010862], 'sidewalk']

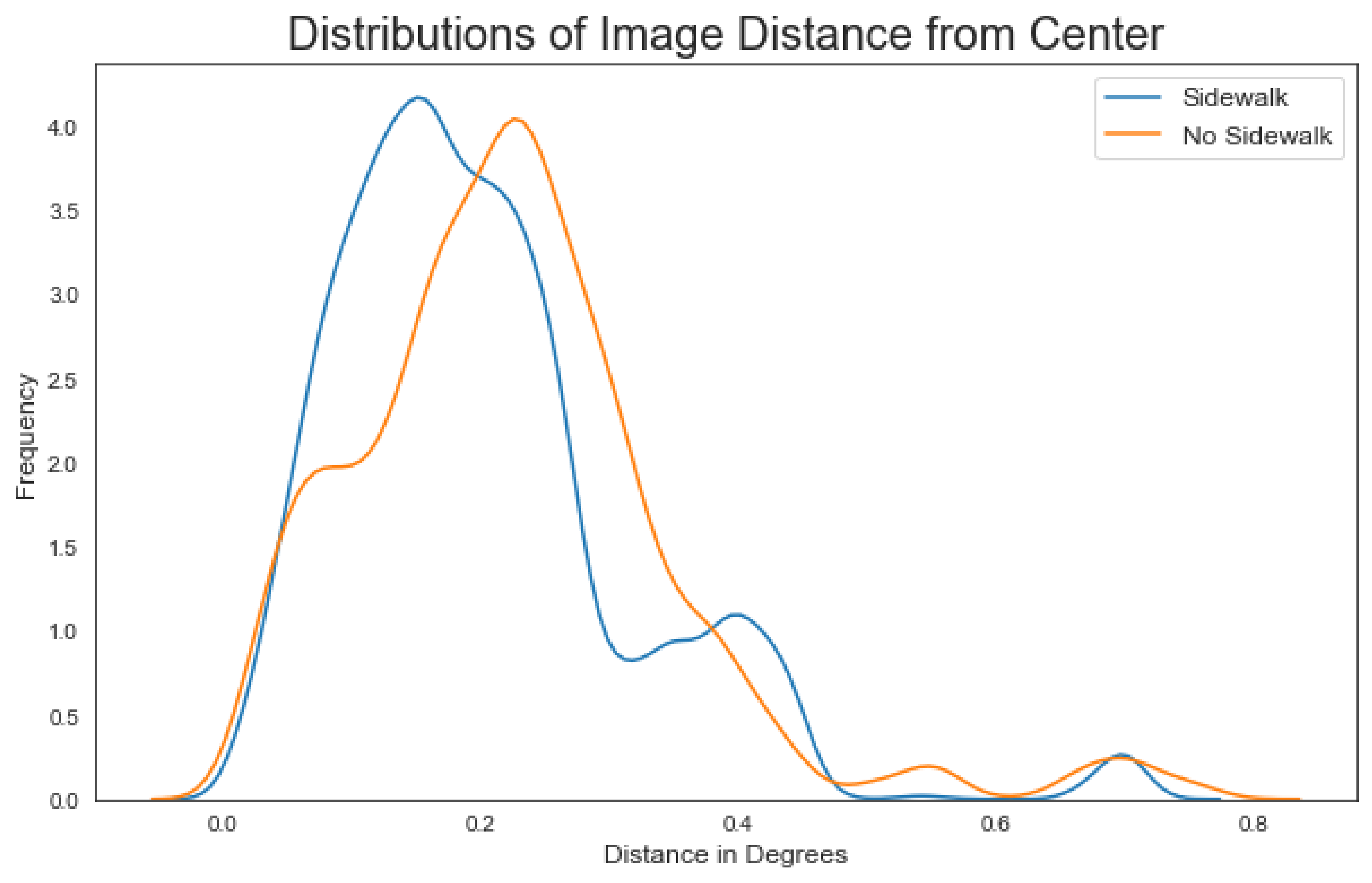
Data Results

- 18,000 Images
- 77% had sidewalks
- Around 15% of the images acquired had incorrect labels. This could be for multiple reasons:
  - new construction in last several years
  - partial sidewalk
  - ambiguous ‘sidewalks’
  - some panoramas from google are rotated up to 90 degrees so may include part of the opposite road

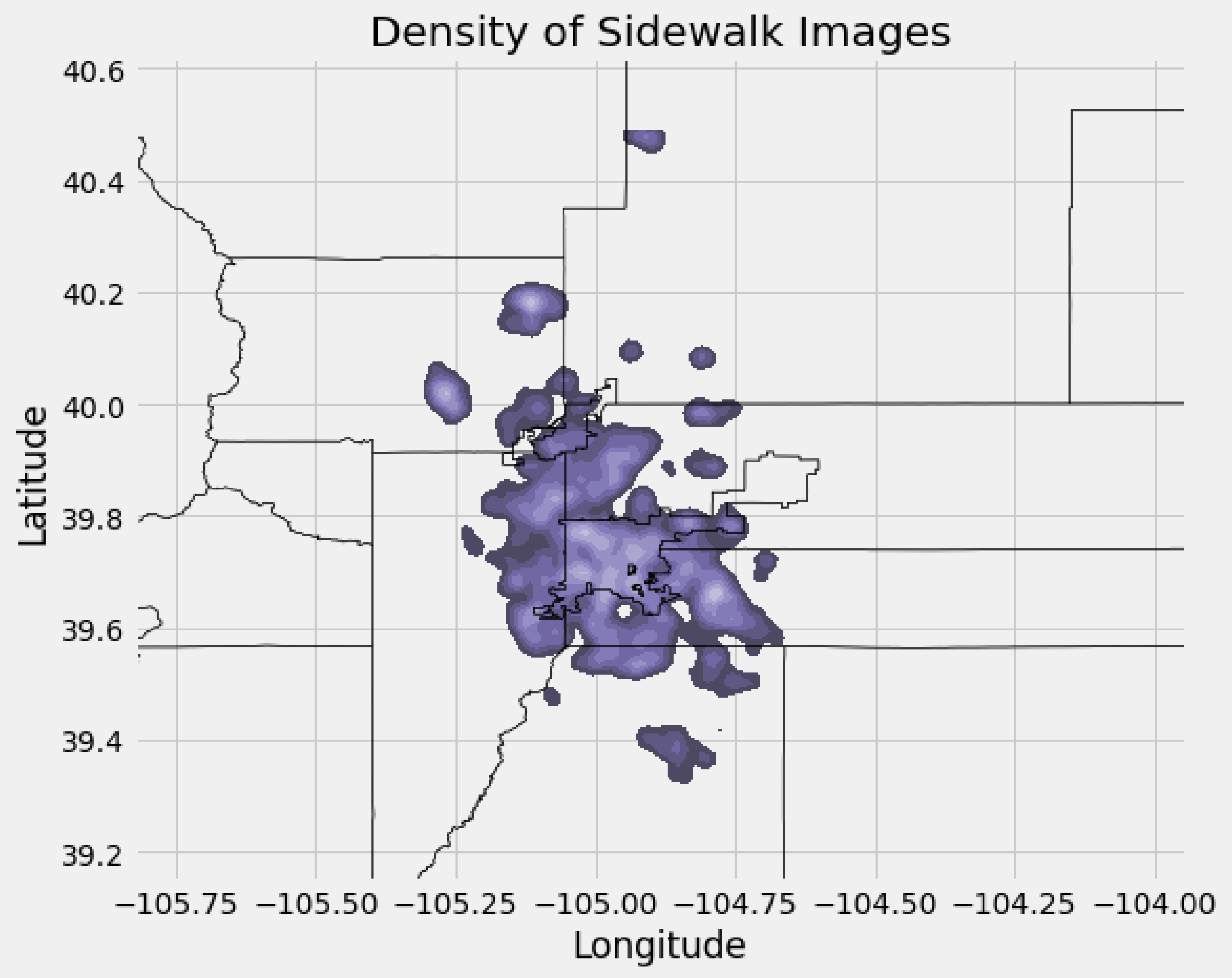
Data Results:

- The images covered most of the region
- Non-sidewalk images generally were from outlying areas
- Non-sidewalk images were less distributed

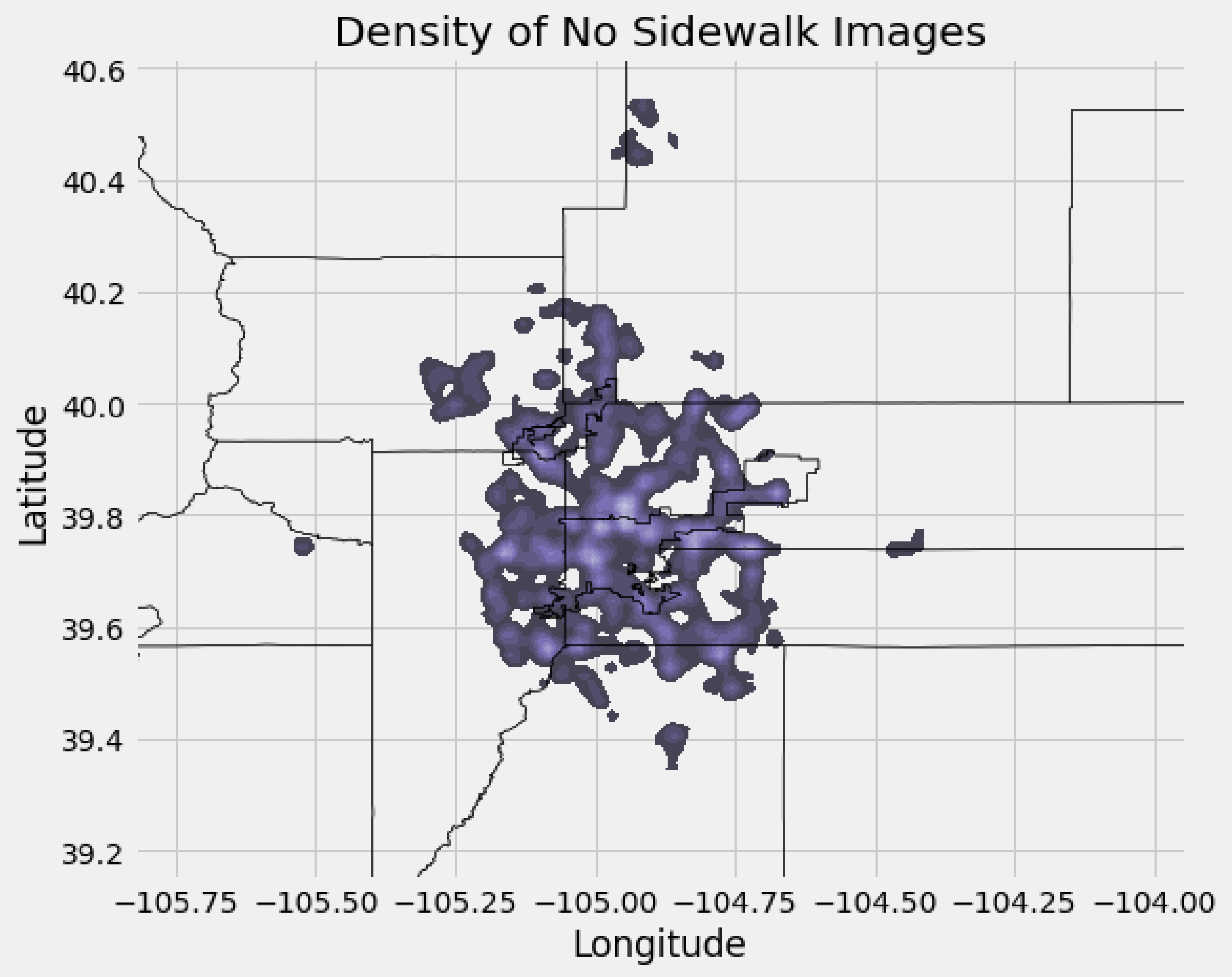
Data Results:



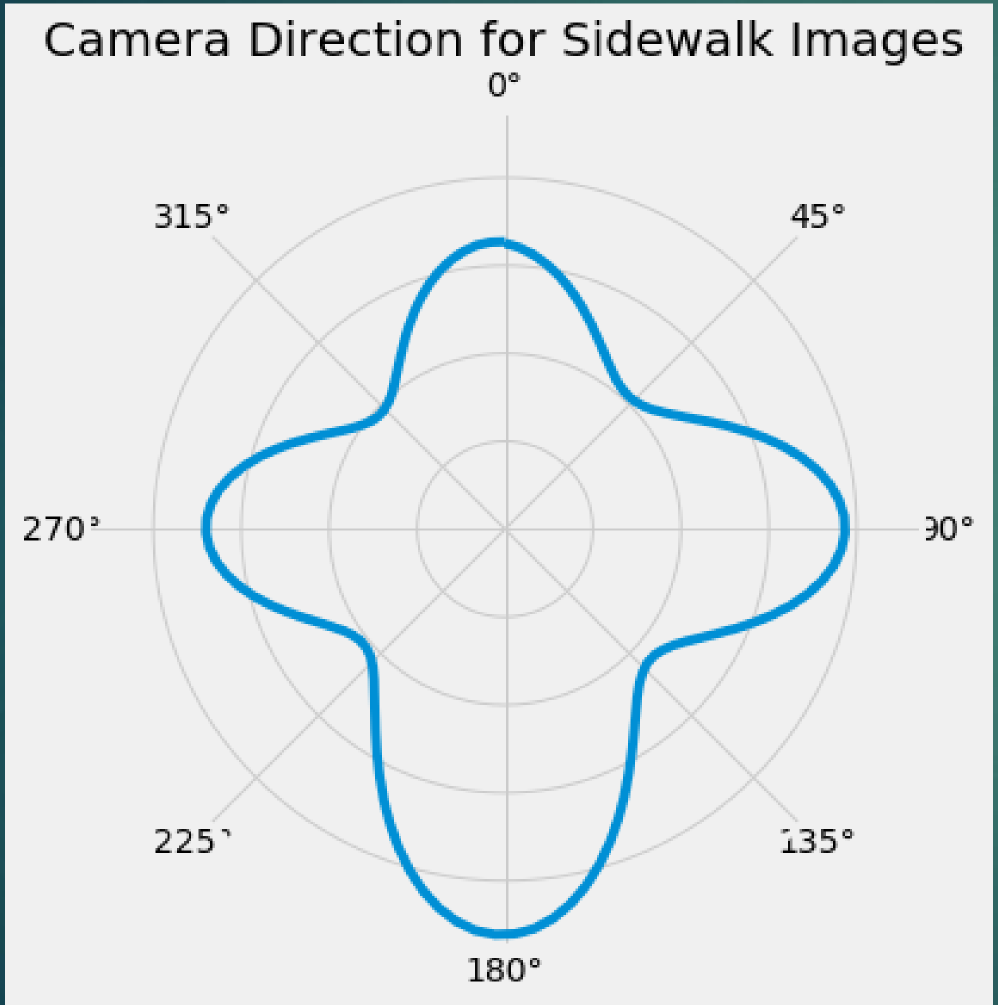
Data Results:

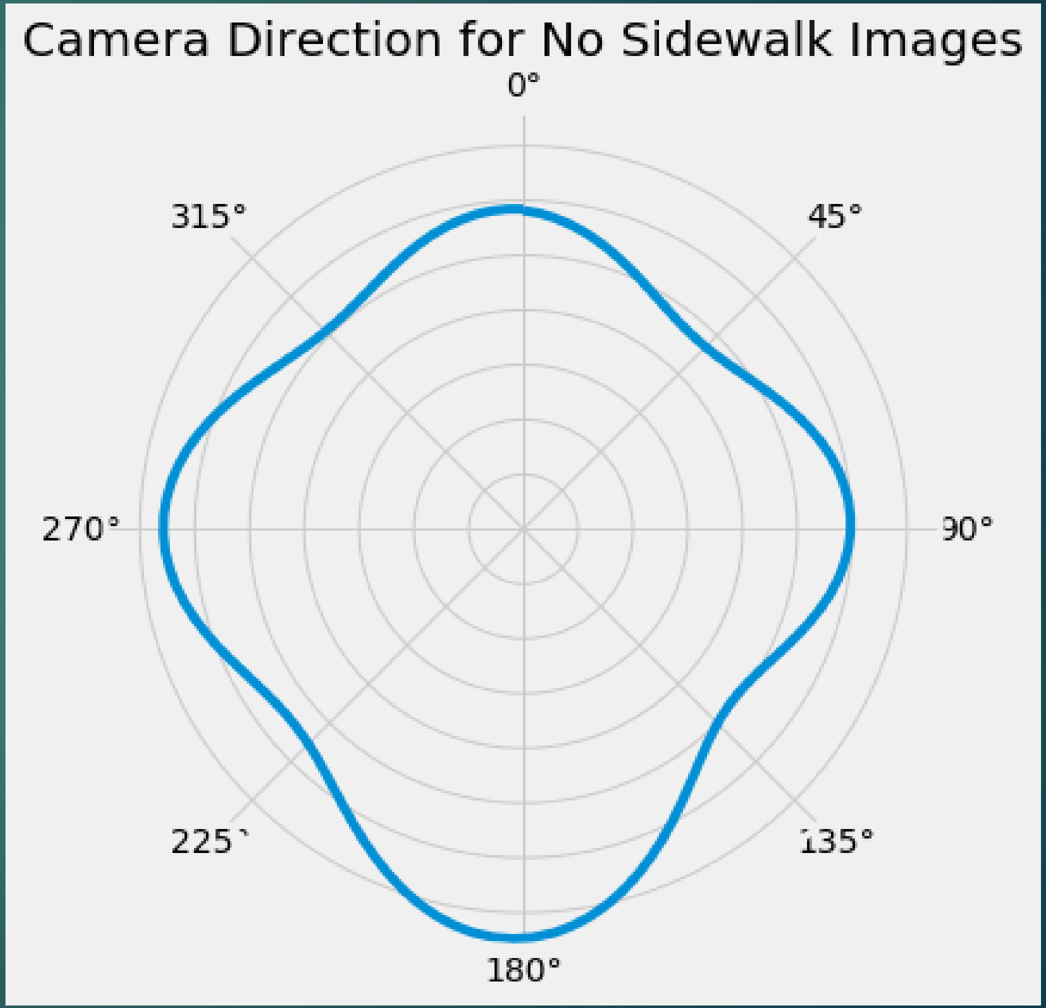


Data Results:



Data Results:

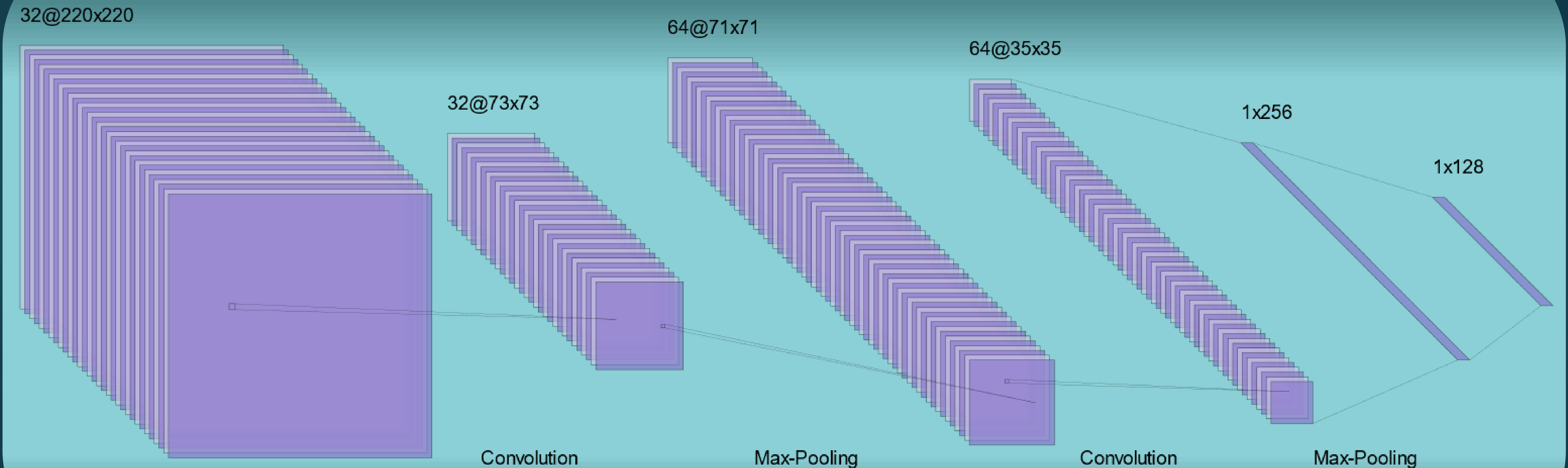




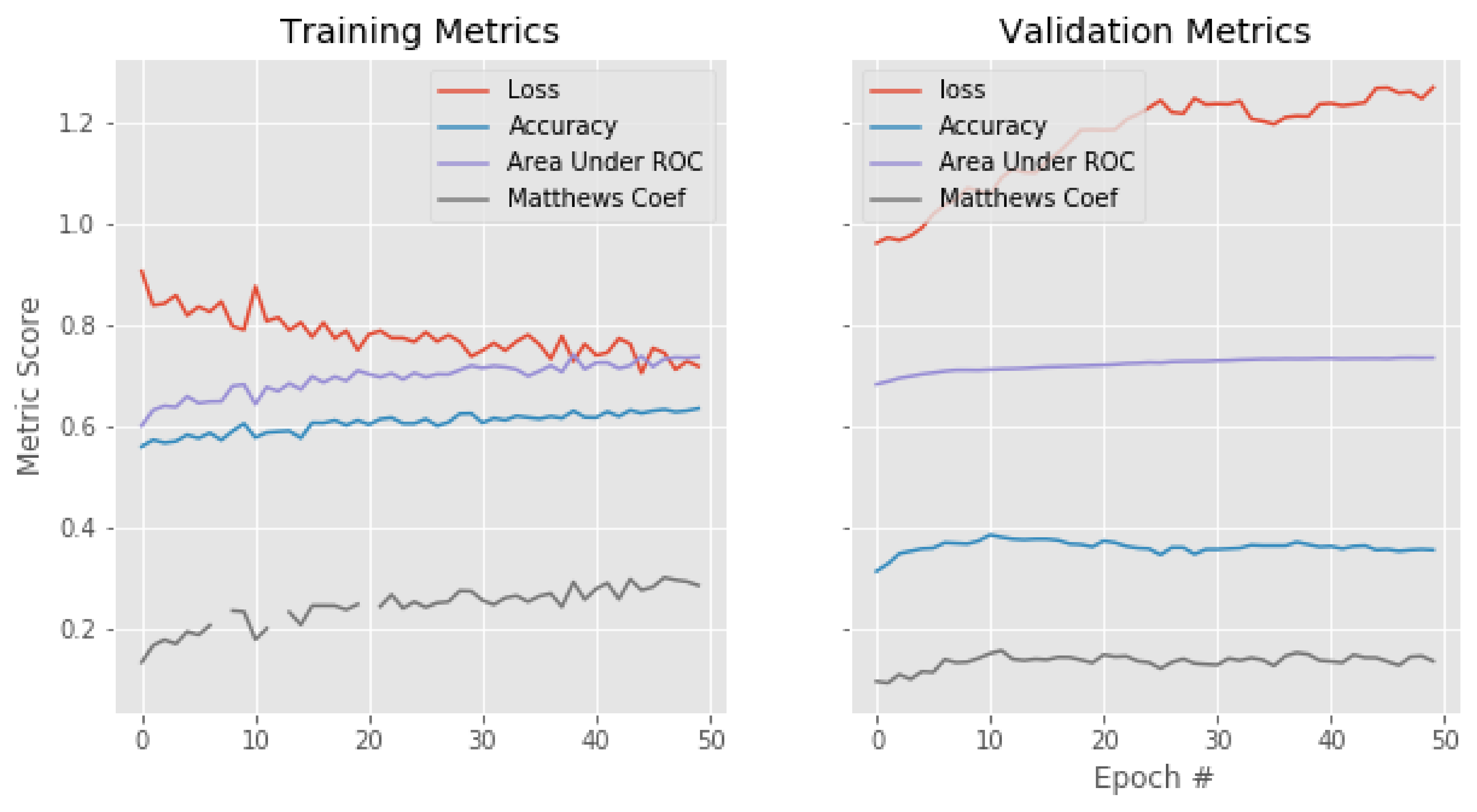
Modelling: Classification

- Use tf.keras to build models:
  - Model with several convolutional layers
- Pretrained models:
  - Mobilenetv3
  - Xception

Modelling: Classification

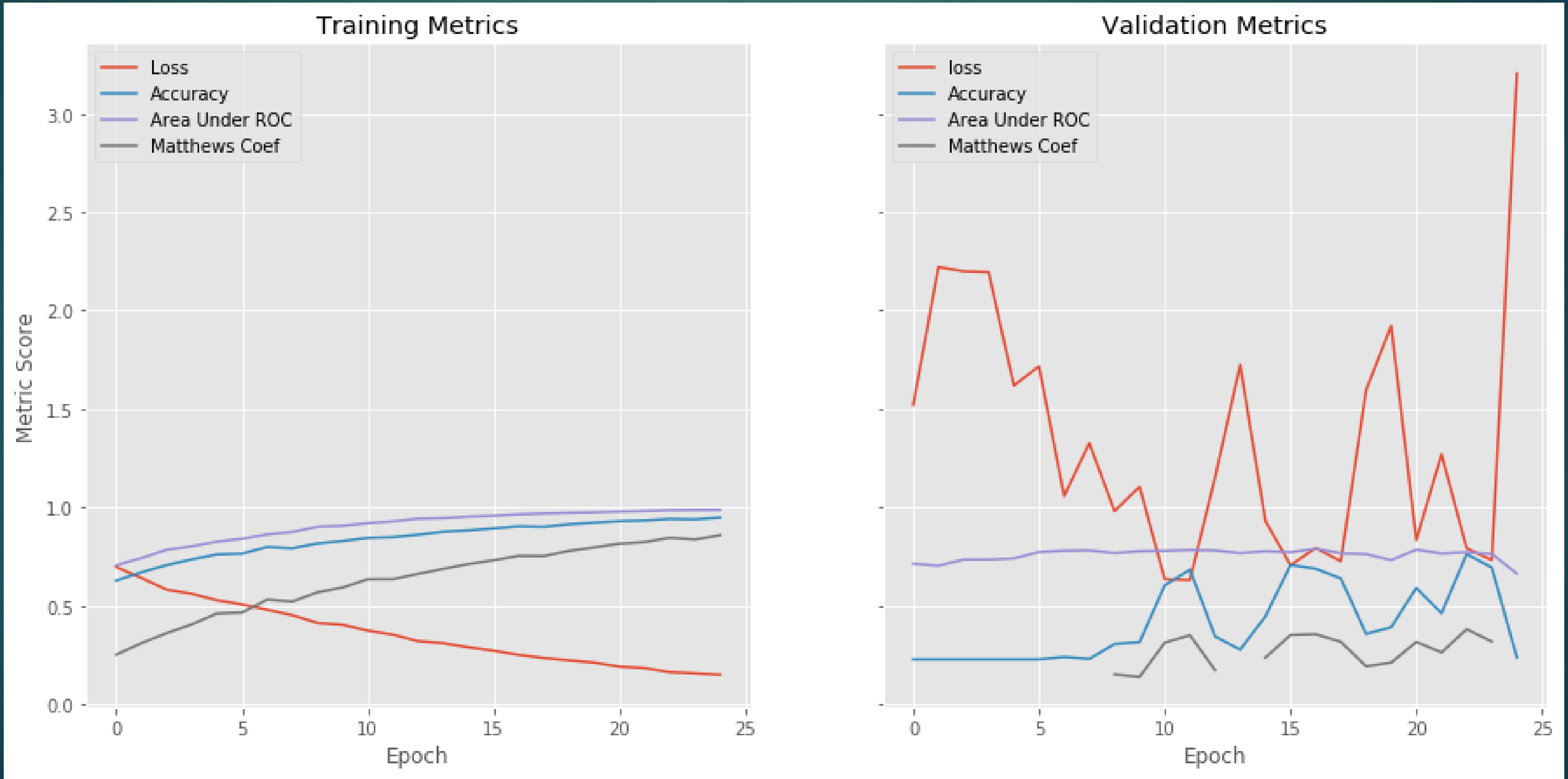


Modelling: Classification



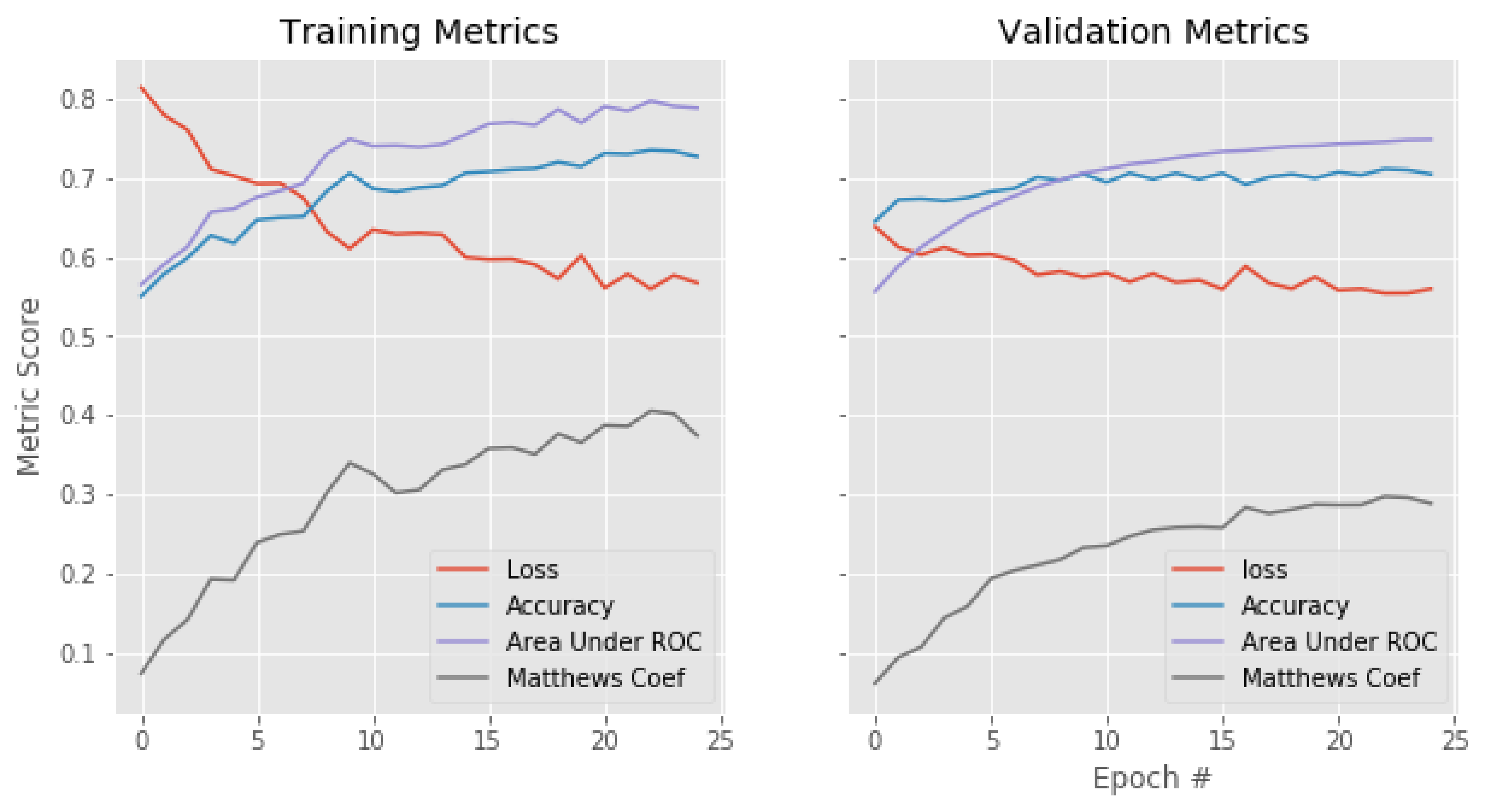
Modelling: Classification

Convolutional Model:



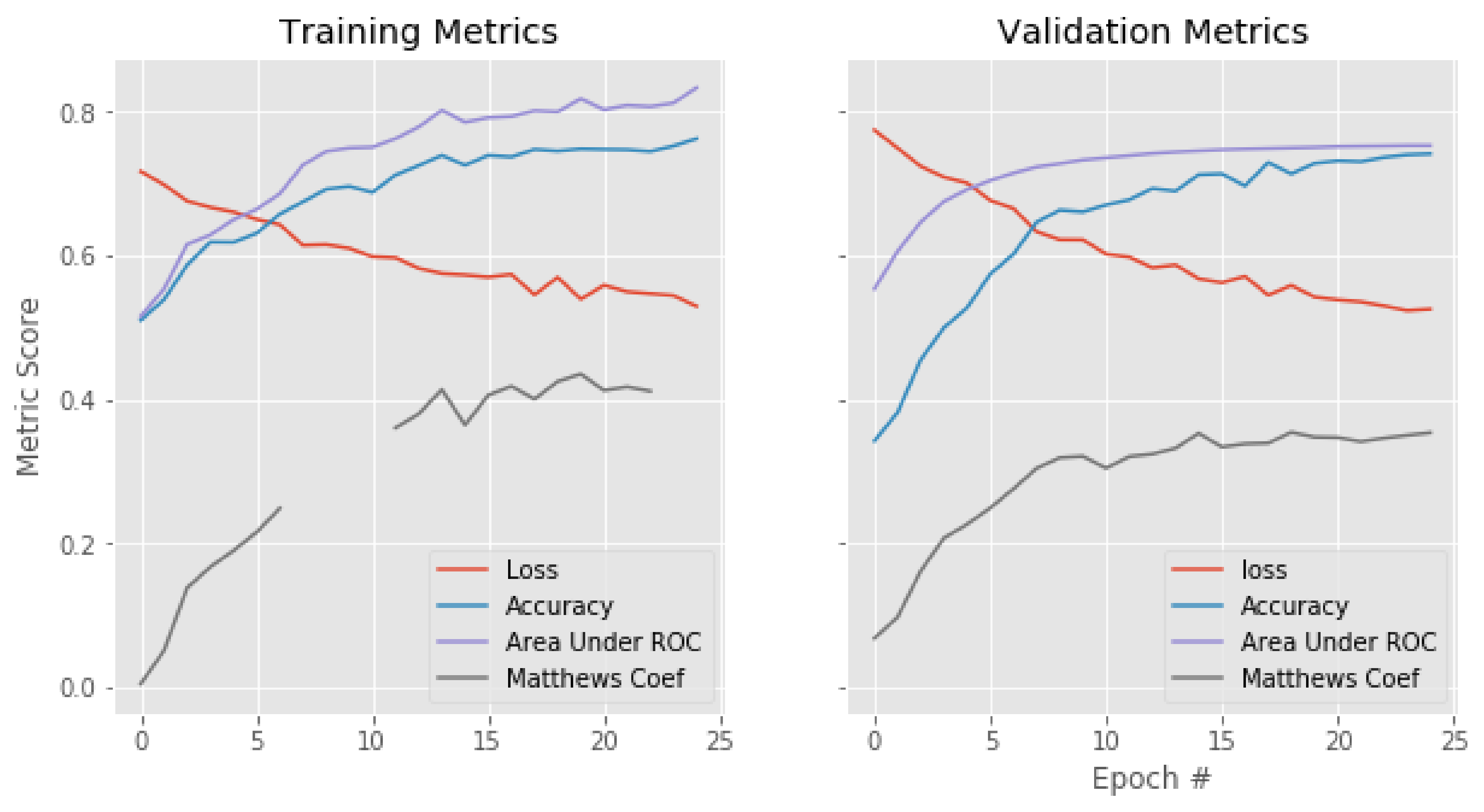
Modelling: Classification

Mobilenetv3



Modelling: Classification

Xception



Modelling: Classification

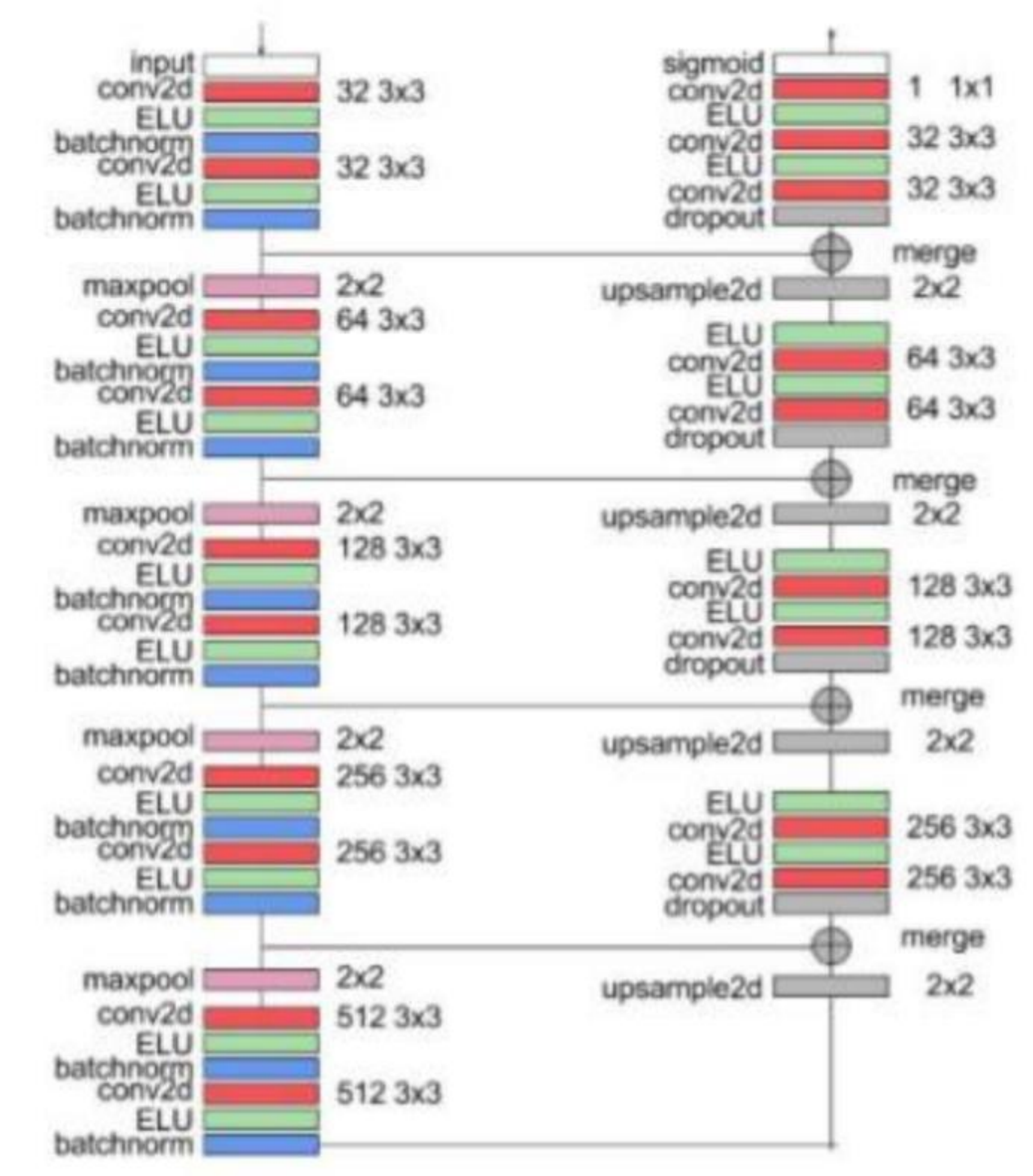
Xception

```
Test results:
Binary Cross-Entropy: 0.5385
Accuracy: 0.7489
Area Under ROC: 0.7803
Matthews Correlation Coefficient: 0.4219
```

Modelling: Image Segmentation

- U-net Model
  - Encoders and Decoders, with skip connection

Architecture Design:



Ref: http://cs230.stanford.edu/files_winter_2018/projects/6937642.pdf

Modelling: : Image Segmentation

Encoder Block:

```
def encoder_builder(input_, filters,
                            activ='relu', kernel=(3,3),
                            drop=.5, pad='same', kern_init='he_uniform'):
  kwargs = {'filters': filters, 'activation': activ, 'kernel_size': kernel,
        'padding': pad, 'kernel_initializer': kern_init}
  x = Conv2D(**kwargs)(input_)
  x = BatchNormalization()(x)
  x = Dropout(drop)(x)
  x = Conv2D(**kwargs)(x)
  encoder = Dropout(drop)(x)
  pooled = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(encoder)
  return encoder, pooled
```

Modelling: Image Segmentation

Decoder Block:

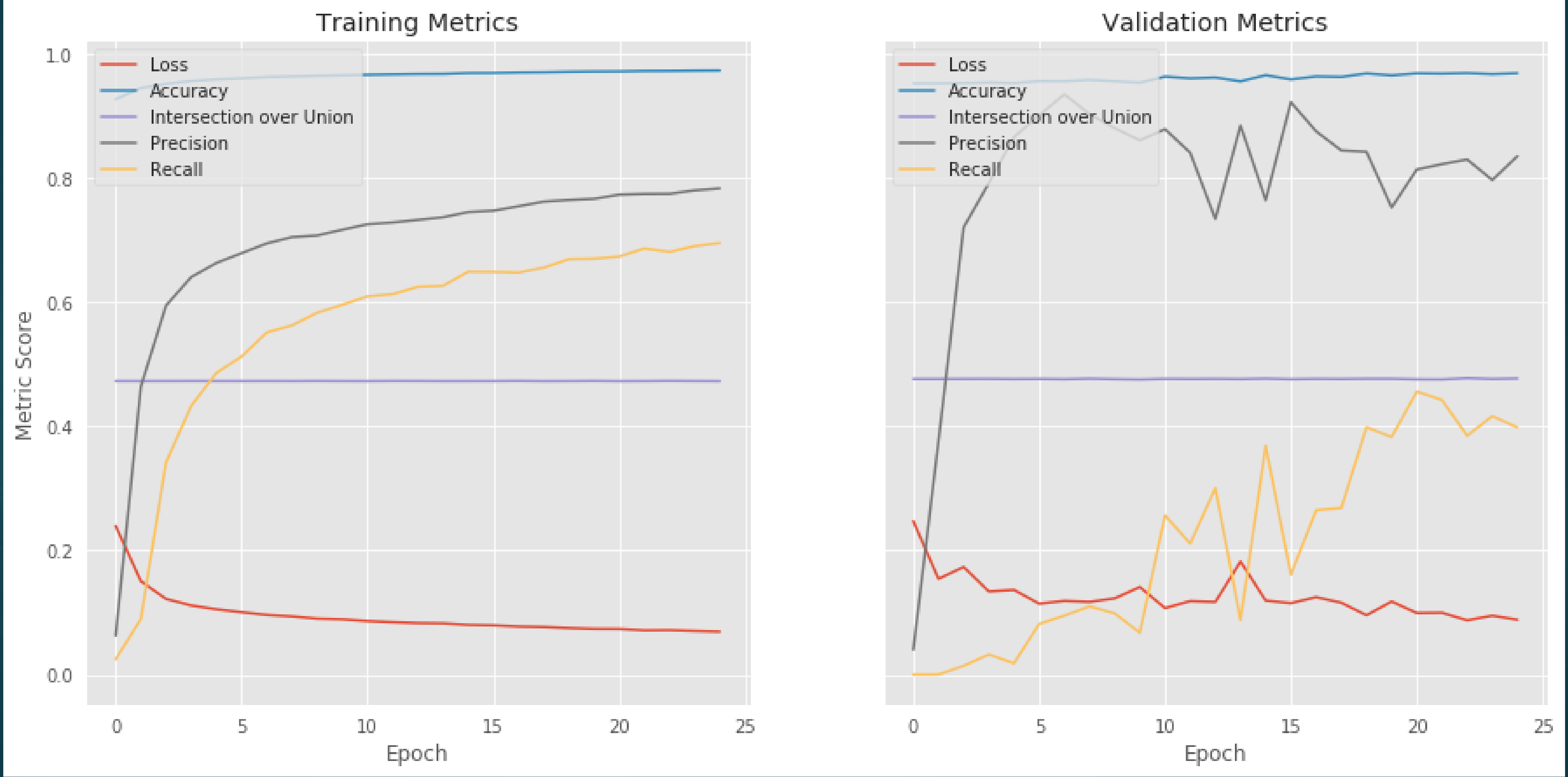
```
def decoder_builder(input_, skip, filters,
                    activ='relu', kernel=(2,2),
                    drop=.5, pad='same', kern_init='he_uniform',
                    ):
  kwargs = {'filters': filters, 'activation': activ, 'kernel_size': kernel,
       'padding': pad, 'kernel_initializer': kern_init}
  x = Conv2DTranspose(**kwargs, strides=(2,2))(input_)
  x = concatenate([x, skip], axis=-1)  # note axis is *-*1
  x = BatchNormalization()(x)
  x = Dropout(drop)(x)
  x = Conv2D(**kwargs)(x)
  x = BatchNormalization()(x)
  x = Dropout(drop)(x)
  x = Conv2D(**kwargs)(x)
  return x
```

Modelling: Image Segmentation

Overall Structure:

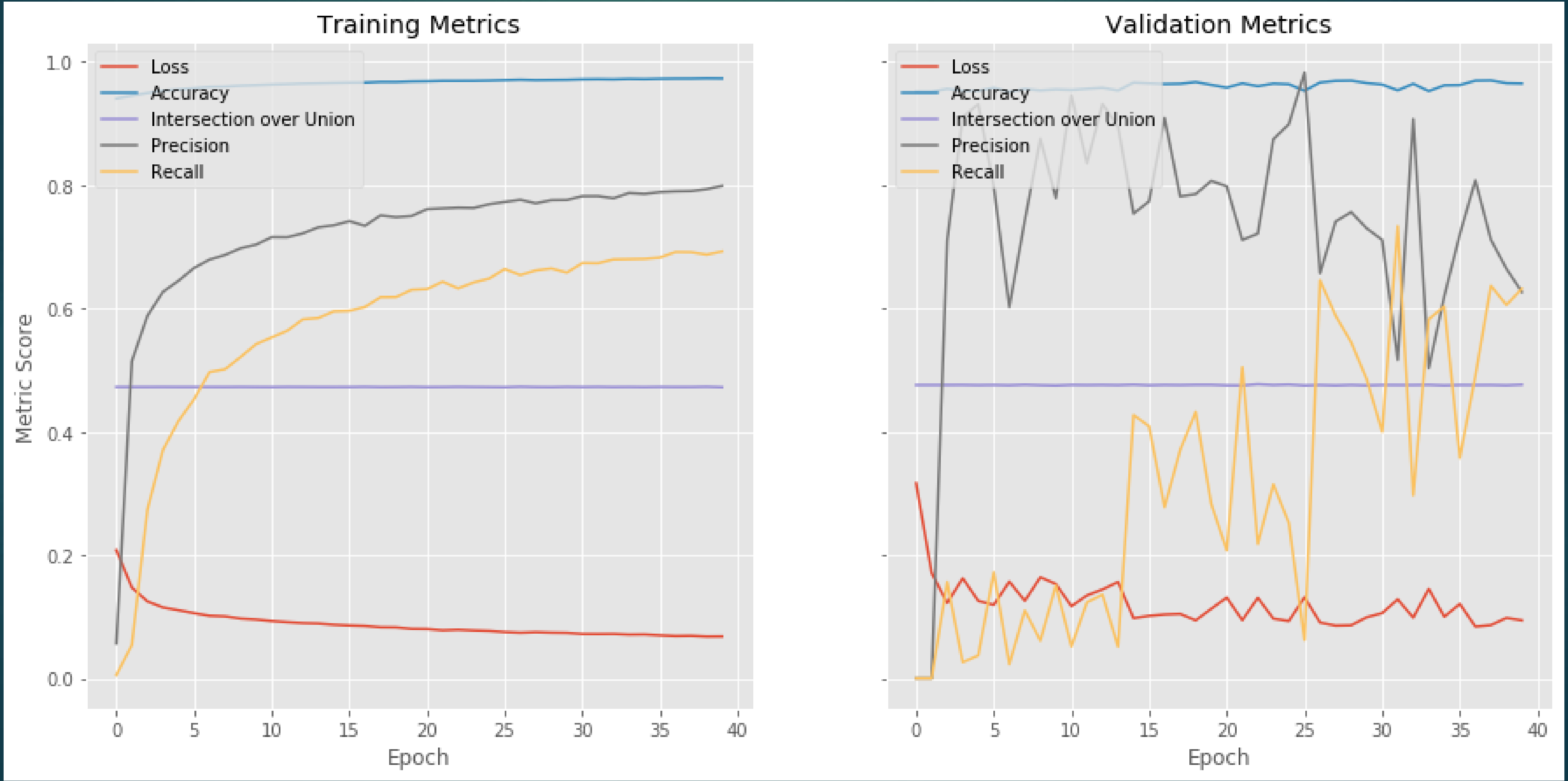
```python
input_layer = Input(shape=img_shape)
encoder1, pooled1 = encoder_builder(input_layer, filters=16)  # return (256x128x32)
encoder2, pooled2 = encoder_builder(pooled1, filters=32)  # return (128x64x64)
encoder3, pooled3 = encoder_builder(pooled2, filters=64)  # return (128x32x128)
encoder4, pooled4 = encoder_builder(pooled3, filters=128)  # return (32x16x256)
middle, middle_pool = encoder_builder(pooled4, filters=128)  # return (16x16x512)
decoder256 = decoder_builder(middle, skip=encoder4, filters=128)
decoder128 = decoder_builder(decoder256, skip=encoder3, filters=64)
decoder64 = decoder_builder(decoder128, skip=encoder2, filters=32)
decoder32 = decoder_builder(decoder64, skip=encoder1, filters=16)
out_layer = Conv2D(filters=1, kernel_size=(1, 1),
                   activation='sigmoid')(decoder32)
```

Modelling:



Training Metrics
Validation Metrics
Loss
Accuracy
Intersection over Union
Precision
Recall
Metric Score
Epoch
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25

Modelling:



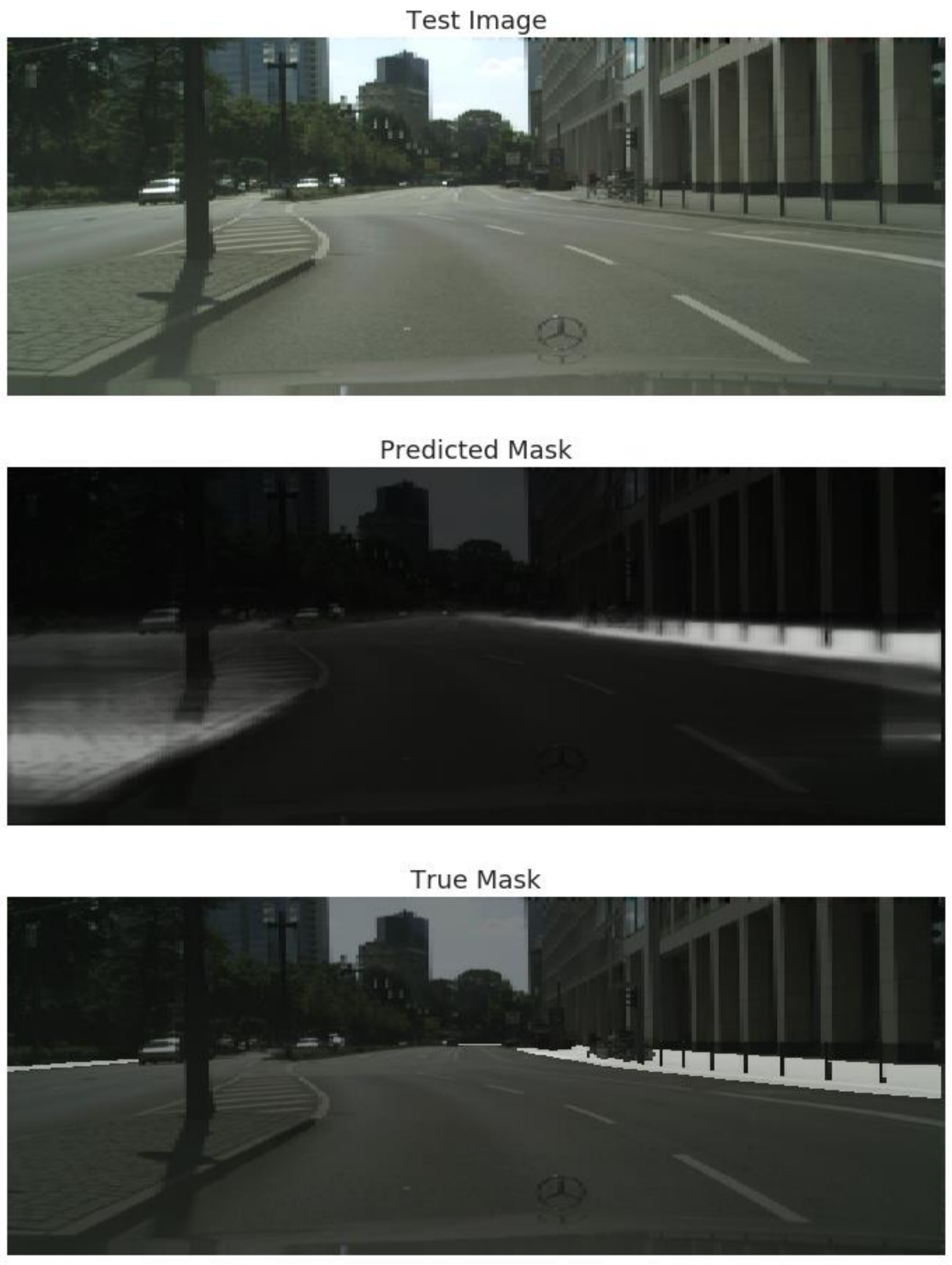
Training Metrics
Validation Metrics
Loss
Accuracy
Intersection over Union
Precision
Recall
Metric Score
Epoch
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
40

Modelling: Image Segmentation

Test Image

Modelling: Image Segmentation

Modelling: Image Segmentation


Test Image
Predicted Mask
True Mask

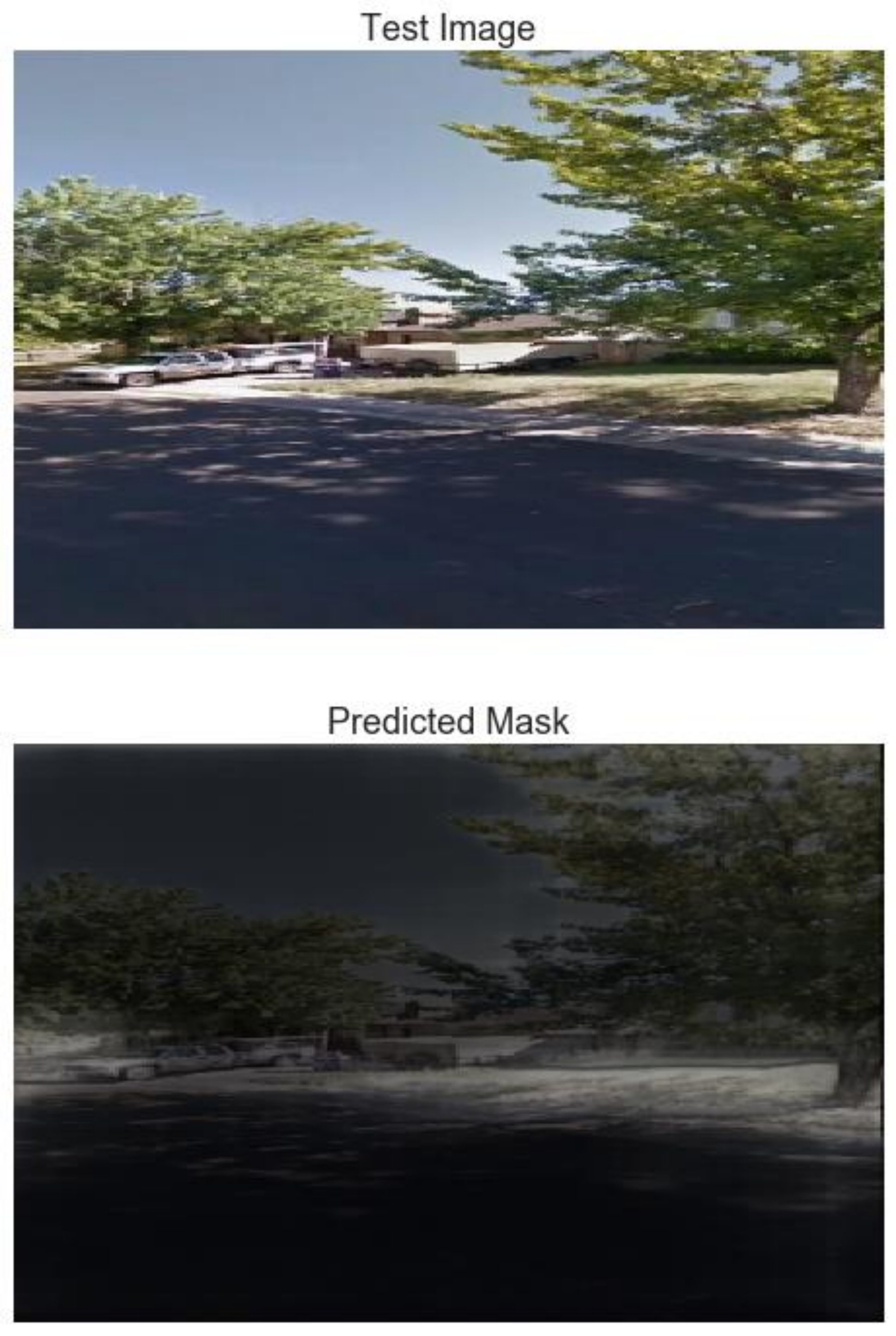

Test Image
Predicted Mask
True Mask

Modelling: Direct Transfer Learning

Use of trained U-net Model on Denver Street Images:

Test Image

Predicted Mask


Test Image
Predicted Mask

Modelling: Direct Transfer Learning

Images without Sidewalks:

Test Image

Predicted Mask

Test Image

Predicted Mask

Known Problems

- Misclassified images
- Mean Intersection over Union measure stuck at 47%
- Find better data augmentation parameters that improve validation scores

Future Directions

- Additional tuning of hyperparameters
- Unfreezing Xception model for additional training
- Additional Data
- Binary segmentation vs probabilistic
- Pretrained backbone for Unet Model
- Segmentation labeling the Denver dataset for additional training
- Using segmentation model to classify
- Reducing code down to CLI script

Questions?