CERBERUS

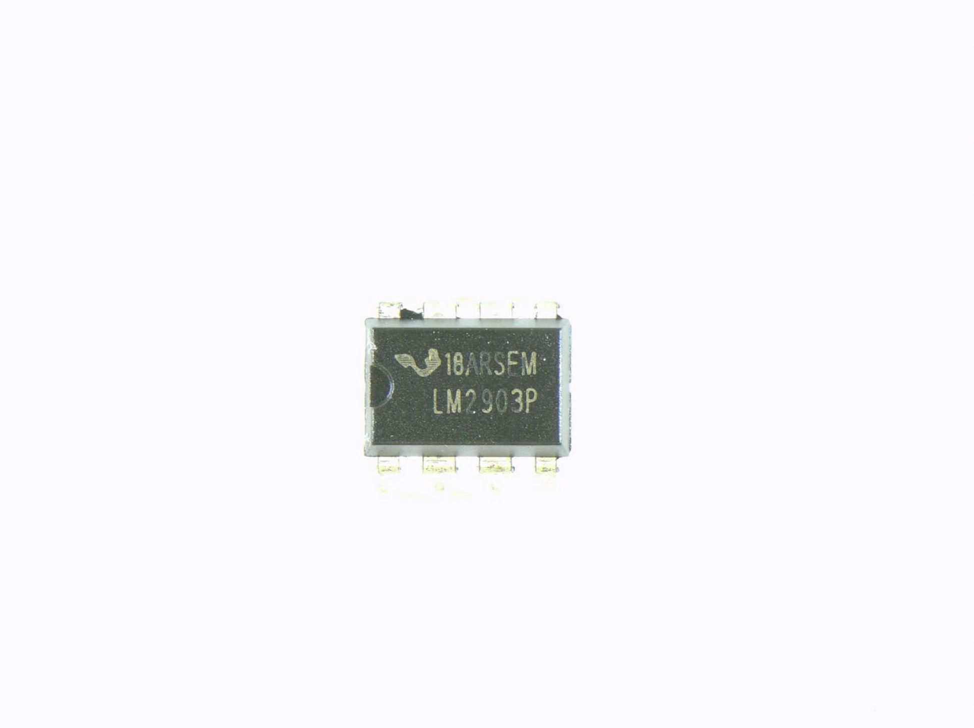
IEEE HOST 2023 - SCS

Andrew MacGillivray, Faith Hedges, Tanvir Hossain, Viet Le, Tazmidul Hoque, and Sumaiya Shomaji
University of Kansas

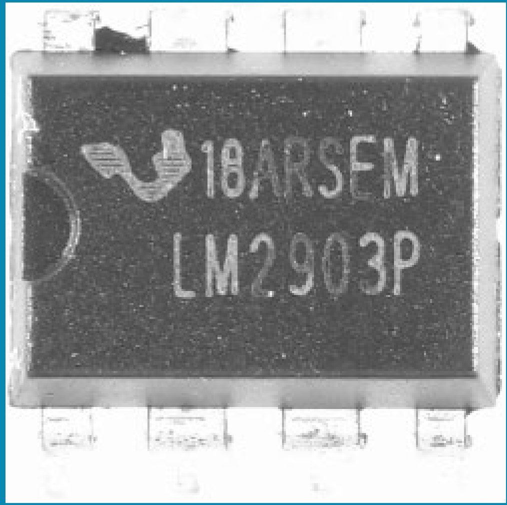
Approach

- Convolutional Neural Network
- Small training set -> short training length
- Significant input image processing to maximize features

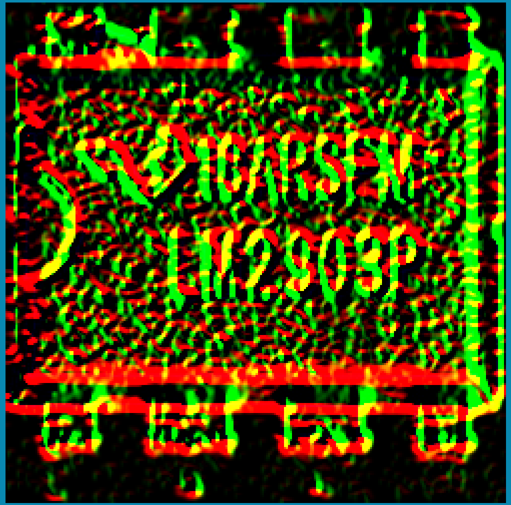
Methodology 1)
Image processing



1. Original Image



2. Whitespace Removal and size reduction



2. Output (feature extraction) – taken from Laplacian (blue), SobelX (green) and SobelY (red)

Andrew MacGillivray, Faith Hedges, Tanvir Hossain, Viet Le, Tazmidul Hoque, and Sumaiya Shomaji
University of Kansas

Methodology 2) **Modeling**

Build Model

The following cells build the model, using the Keras Model class.

```
In [7]:
inputs = Input(shape=(IMAGE_SIZE + (3,)))

n = 8
b1a = "elu"
b2a = b1a
b3a = b1a


x = Conv2D(n, (3, 3), activation = b1a)(inputs)
x = BatchNormalization()(x)
x = Conv2D(n*2, (3, 3), activation = b1a)(x)
block_1_output = MaxPool2D(pool_size=(3, 3))(x)

x = Conv2D(n*2, (3, 3), activation = b2a, padding = 'same')(block_1_output)
x = BatchNormalization()(x)
x = Conv2D(n*2, (3, 3), activation = b2a, padding = 'same')(x)
block_2_output = add([x, block_1_output])

x = Conv2D(n*2, (3, 3), activation = b3a, padding = 'same')(block_2_output)
x = BatchNormalization()(x)
x = Conv2D(n*2, (3, 3), activation = b3a, padding = 'same')(x)
block_3_output = add([x, block_2_output])

x = Conv2D(n*2*2, (3, 3), activation = 'elu')(block_3_output)
x = MaxPool2D(pool_size = (2, 2))(x)
x = GlobalAveragePooling2D()(x)
x = Dense(n*2*2*2, activation = 'elu')(x)

output = Dense(1, activation = 'sigmoid')(x)

if DEBUG:
    print(inputs.shape)

In [8]:
model = Model(inputs, output)

In [9]:
model.compile(metrics=METRICS, loss=LOSS, optimizer=OPTIMIZER)
model.summary()

Model: "model"
_____________________________________________________________________________
 Layer (type)                    Output Shape         Param #     Connected to
=============================================================================
```

1. Model definition

```
=============================================================================
Total params: 17,649
Trainable params: 17,569
Non-trainable params: 80
_____________________________________________________________________________

[10]:
history = model.fit(
    train_images,
    train_labels,
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
    validation_data=(test_images,test_labels),
    shuffle=True
)

Epoch 1/20
9/9 [==============================] - 13s 1s/step - loss: 0.7720 - Accuracy: 0.4767 - val_loss: 1.6083 -
Epoch 2/20
9/9 [==============================] - 10s 1s/step - loss: 0.6697 - Accuracy: 0.5814 - val_loss: 0.9450 -
Epoch 3/20
9/9 [==============================] - 10s 1s/step - loss: 0.6270 - Accuracy: 0.6512 - val_loss: 0.9913 -
Epoch 4/20
9/9 [==============================] - 10s 1s/step - loss: 0.5999 - Accuracy: 0.6977 - val_loss: 0.7667 -
Epoch 5/20
9/9 [==============================] - 9s 1s/step - loss: 0.5751 - Accuracy: 0.6744 - val_loss: 0.9935 -
Epoch 6/20
9/9 [==============================] - 9s 1s/step - loss: 0.5685 - Accuracy: 0.7558 - val_loss: 0.8247 -
Epoch 7/20
9/9 [==============================] - 9s 1s/step - loss: 0.5028 - Accuracy: 0.8140 - val_loss: 1.2283 -
Epoch 8/20
```

2. Training

We ran until we yielded a model with similar accuracy on both the training and validation sets with reasonable loss values. Our submitted model had accuracy in the 60-65% range for both sets.

Andrew MacGillivray, Faith Hedges, Tanvir Hossain, Viet Le, Tazmidul Hoque, and Sumaiya Shomaji
University of Kansas

Results

The trained model is tested; accuracy is measured on its predictions for both the training and test (validation) sets.

Our final model predicts 92 of 120 labels correctly, achieving a total accuracy of ~76%

Test Confidence Filter on Training Data

Here, we'll apply our extra post-processing of the predictions and see how accurat

```
In [12]:
# Ensure that all images - including those we excluded from the traini
USE_EXCLUDE = False
(ci_all, cl_all), (gi_all, gl_all) = load_data(training_path,["counter
cf_iset = np.append(ci_all, gi_all, axis=0) # Confidence Filter - Imag
cf_lset = np.append(cl_all, gl_all, axis=0) # Confidence Filter - Labe

Setting DIRS to CATEGORY...
Loading counterfeit
Applying label:  1
Loading genuine
Applying label:  0
Error: Unable to read file ' ../input/host-23/phase1-workspace/genuine/

In [13]:
predict_with_confidence_knownvals( model, cf_iset, cf_lset )

4/4 [==============================] - 3s 584ms/step
Correct Predictions:  75
Total Predictions:  100
Accuracy:  0.75
Confusion:
tf.Tensor(
[[60  0]
 [25 15]], shape=(2, 2), dtype=int32)
Authentics Predicted:  85
Counterfeits Predicted:  15

In [14]:
predict_with_confidence_knownvals( model, test_images, test_labels )

1/1 [==============================] - 1s 521ms/step
Correct Predictions:  13
Total Predictions:  20
Accuracy:  0.65
Confusion:
tf.Tensor(
[[10  0]
 [ 7  3]], shape=(2, 2), dtype=int32)
Authentics Predicted:  17
Counterfeits Predicted:  3
```

Note: this is not from the run that produced our final model

```
# Predict labels of a data set, then apply a c
# -----
# Args:
#  m:    model        (Keras Model)
#  ds:   data set     (Input)
def predict_with_confidence( m, ds ):
    predictions = model.predict(x=ds)
    filtered_preds = []
    if DEBUG:
        print(predictions)
    for idx, pred in enumerate(predictions) :
        if (pred <= CONF_T):
            filtered_preds.append(0)
        else:
            filtered_preds.append(1)
    return filtered_preds

# Predict labels of a data set, then apply a c
# Compare predictions with the known labels of
# -----
# Args:
#  m:    model        (Keras Model)
#  ds:   data set     (Input)
#  dl:   data labels
def predict_with_confidence_knownvals( m, ds,
    filtered_preds = predict_with_confidence(m
    label_length = len(dl)
    correct_preds = 0
    auth_preds = 0
    cfit_preds = 0
```

Training Set Confusion		
	True Authentic	True Counterfeit
Predicted Authentic	60	23
Predicted Counterfeit	0	17
Accuracy:	77%	

Test Set Confusion		
	True Authentic	True Counterfeit
Predicted Authentic	9	4
Predicted Counterfeit	1	6
Accuracy:	75%	

Andrew MacGillivray, Faith Hedges, Tanvir Hossain, Viet Le, Tazmidul Hoque, and Sumaiya Shomaji
University of Kansas

Discussion

- Model produces more false-negatives than desired
- Could be improved with:
  - More data
  - Better image preprocessing
  - More tinkering with the model
  - Creation of more specific models (e.g. front/back of chip)

Andrew MacGillivray, Faith Hedges, Tanvir Hossain, Viet Le, Tazmidul Hoque, and Sumaiya Shomaji
University of Kansas