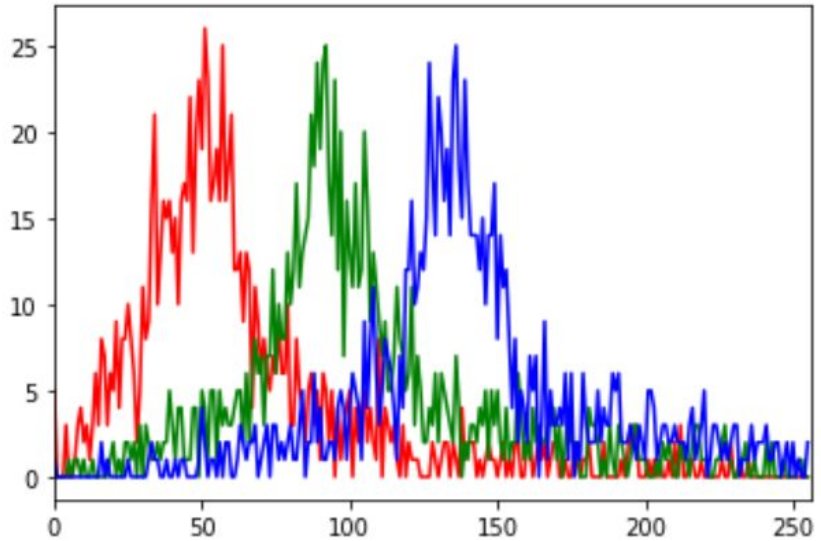
ADS Project 3

Weakly supervised learning -- label noise and correction

Ying Gao, Alix Leon, Shreya Sinha, Weijia Wang, Tomasz Wislicki

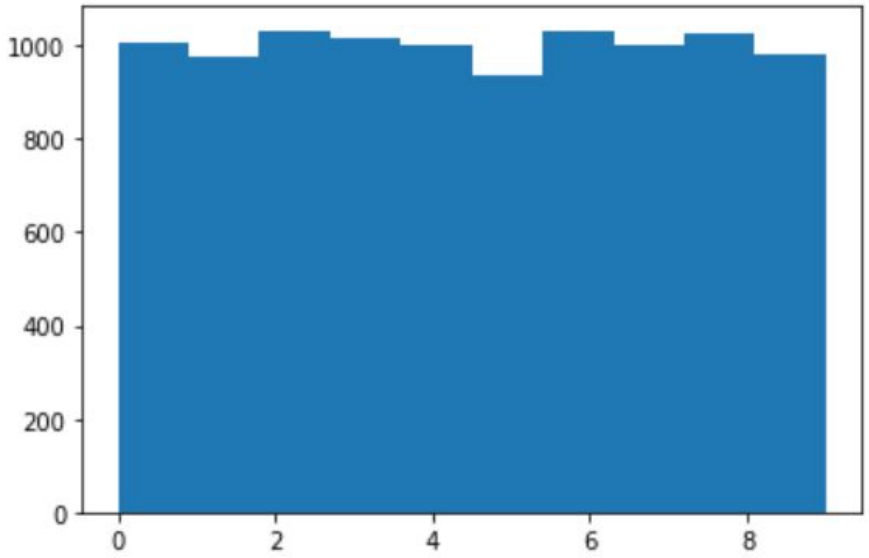
Data Preprocessing

Dimensions: (32, 32, 3)



```
plt.hist(clean_labels)
```

```
(array([1005.,  974., 1032., 1016.,  999.,  937., 1030., 1001., 1025.,
         981.]),
 array([0. , 0.9, 1.8, 2.7, 3.6, 4.5, 5.4, 6.3, 7.2, 8.1, 9. ]),
 )
```



Model Selection

Baseline Model: Logistic Regression

Model1: ResNet18

Model2: ResNet18 + Image Classifier + Label Cleaner

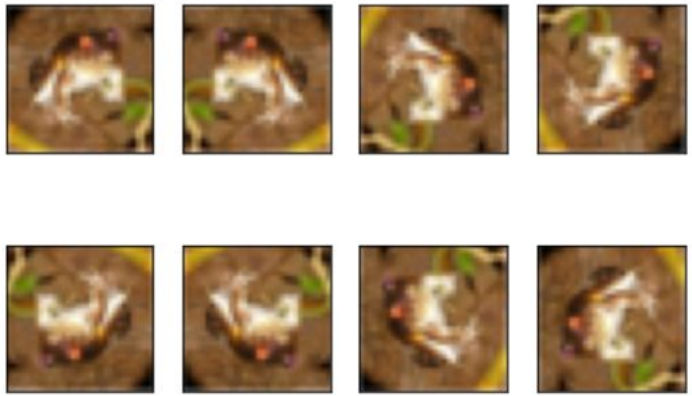
Baseline Model: Logistic Regression

```
evaluation(baseline_model, clean_labels, imgs[:10000])
```

	precision	recall	f1-score	support
0	0.32	0.43	0.37	1005
1	0.18	0.29	0.22	974
2	0.22	0.04	0.07	1032
3	0.19	0.12	0.14	1016
4	0.24	0.48	0.32	999
5	0.22	0.13	0.16	937
6	0.26	0.35	0.30	1030
7	0.29	0.04	0.07	1001
8	0.28	0.43	0.34	1025
9	0.19	0.11	0.14	981
accuracy			0.24	10000
macro avg	0.24	0.24	0.21	10000
weighted avg	0.24	0.24	0.21	10000

Data Augmentation

```
test_images = generate_rotations(images[0])
for i, img in enumerate(test_images):
    plt.subplot(2, 4, i + 1)
    plt.imshow(cv2.resize(img, (32, 32)) / 255.0)
    plt.xticks([]) ## remove the ticks on x-axis
    plt.yticks([]) ## remove the ticks on y-axis
plt.show()
```



Model Selection: Model 1

Model	Number of Epochs trained	Accuracy
CNN	58	0.6300
ResNet18	44	0.48
EfficientNet	20	0.2227
MobileNet	20	0.1019
VGG16	10	0.2539
ResNet101	10	0.1433
AlexNet	10	0.1780

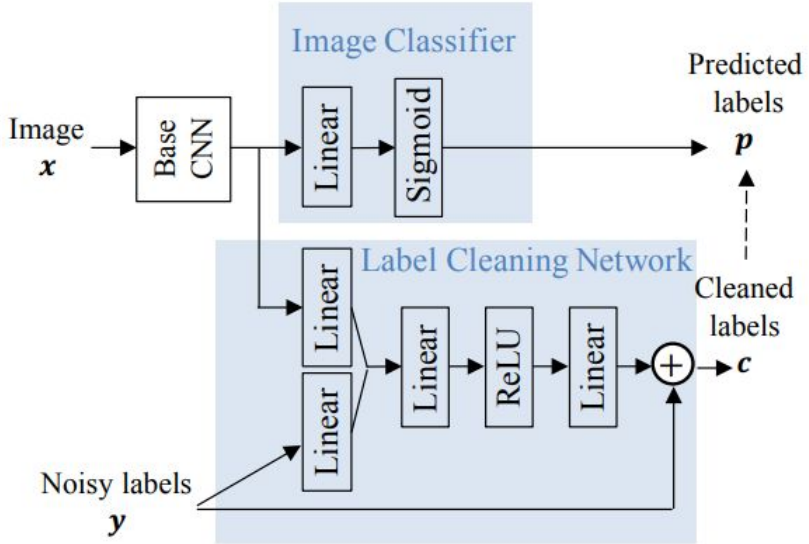
```python
# Model 1
cnn_1 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3,3),
                           strides=1, padding='same',
                           input_shape=(32,32,3), use_bias=True),
    # keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2, padding='valid'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3),
                          strides=1, padding='same', use_bias=True),
    # keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2, padding='valid'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Conv2D(filters=128, kernel_size=(3,3),
                          strides=1, padding='same', use_bias=True),
    tf.keras.layers.Activation('relu'),
    # tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2, padding='valid'),
    # tf.keras.layers.Dropout(0.2),
    tf.keras.layers.GlobalAvgPool2D(),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, use_bias=False),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

```
28/28 [==============================] - 3s 119ms/step - loss: 0.5373 - accuracy: 0.8071 - val_loss: 0.9863 - val_accuracy: 0.6961
Epoch 100/100
28/28 [==============================] - 3s 123ms/step - loss: 0.5235 - accuracy: 0.8096 - val_loss: 0.9980 - val_accuracy: 0.6861
```

```python
cnn_1.evaluate(x_clean_test, y_clean_test)
```

```
32/32 [==============================] - 0s 10ms/step - loss: 1.0284 - accuracy: 0.6650
[1.028435468673706, 0.6650000214576721]
```

Model Selection: Model 2



Reference: [Multi-Label Fashion Image Classification with Minimal Human Supervision]

Model Selection: Model 1 Update - ResNet18

```
[39] from resnet import load_resnet, ResNet18
     ResNet18.trainable=False

[40] resnet_18=ResNet18(10)
     resnet_18.build(input_shape=(None,32,32,3))

[41] resnet_18.compile(loss = tf.keras.losses.CategoricalCrossentropy(),
         optimizer=tf.keras.optimizers.Adam(0.01),
         metrics=['accuracy'])
```

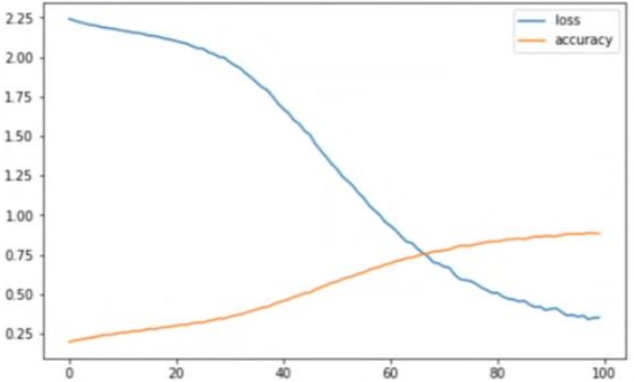
```
Epoch 42/100
157/157 [==============================] - 23s 142ms/step - loss: 1.6437 - accuracy: 0.4676
Epoch 43/100
157/157 [==============================] - 22s 140ms/step - loss: 1.5992 - accuracy: 0.4822
Epoch 44/100
```

```
resnet_18.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 3s 10ms/step - loss: 3.6190 - accuracy: 0.3103
[3.618980884552002, 0.3102999925134033]
```

```
pd.DataFrame(resn_model.history).plot(figsize=(8,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe222298710>
```



Model Selection: Label Cleaner

```python
class LabelCleaner(tf.keras.Model):
    def __init__(self, CNN: tf.keras.Model):
        super(LabelCleaner, self).__init__()

        # Base CNN model
        self.CNN = CNN

        # Fully connected dense layers
        self.fc_1 = tf.keras.layers.Dense(units = 20, use_bias=False)
        self.fc_2 = tf.keras.layers.Dense(units = 256)
        self.fc_3 = tf.keras.layers.Dense(units = 256, use_bias=False, activation = "relu
        self.fc_4 = tf.keras.layers.Dense(units = 10, use_bias=False,)

        # Batch Normalization layers
        self.bn_1 = tf.keras.layers.BatchNormalization()
        self.bn_2 = tf.keras.layers.BatchNormalization()
        self.bn_3 = tf.keras.layers.BatchNormalization()

    def call(self, inputs):
        img, y = inputs
        # Get the CNN output
        x = self.CNN(img)
        # Embed the output of the CNN to the noisy labels
        x = tf.concat([x, y], axis = 1)
        x = self.fc_1(x)    # Linear followed by batch normalization
        x = self.bn_1(x)
        x = self.fc_2(x)    # Linear followed by batch normalization
        x = self.bn_2(x)
        x = self.fc_3(x)    # ReLU
        x = self.fc_4(x)    # Linear followed by batch normalization
        x = self.bn_3(x)
        x = x + y           # Residual connection
        x = tf.clip_by_value(x, 0, 1)
        return x
```

```
Epoch 58/60
57/57 [==============================] - 1s 13ms/step - loss: 14.5798 - accuracy: 0.9425 - val_loss: 18.2836 - val_accuracy: 0.9289
Epoch 59/60
57/57 [==============================] - 1s 13ms/step - loss: 14.5435 - accuracy: 0.9432 - val_loss: 18.0208 - val_accuracy: 0.9278
Epoch 60/60
57/57 [==============================] - 1s 14ms/step - loss: 14.4863 - accuracy: 0.9436 - val_loss: 18.0674 - val_accuracy: 0.9317
<keras.callbacks.History at 0x7f08fae76ad0>

[64] cleaner.evaluate(V_test)
     cleaner.trainable = False

     8/8 [==============================] - 0s 18ms/step - loss: 18.5010 - accuracy: 0.9230
```

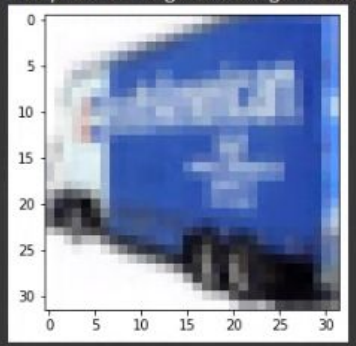
Model Selection: Image Classifier

```python
class ImageClassifier(tf.keras.Model):
    """
    This model is used to classify the images.
    It uses the output from the LabelCleaner model
    as input.

    Parameters:
    -----------
    CNN: tf.keras.Model
        The base CNN model used to extract features from the
        images.
    """

    def __init__(self, cnn: tf.keras.Model):
        super(ImageClassifier, self).__init__()

        self.cnn = cnn
        self.fc_1 = tf.keras.layers.Dense(units = 512),
        self.fc_2 = tf.keras.layers.Dense(units = 10, activation = "sigmoid")

    def call(self, inputs):
        x = self.cnn(inputs)
        x = self.fc_1(x)
        x = self.fc_2(x)

        return x
```

```
Epoch 13/30
57/57 [==============================] - 1s 13ms/step - loss: 0.2912 - accuracy: 0.9296 - val_loss: 0.3209 - val_accuracy: 0.9222
Epoch 14/30
57/57 [==============================] - 1s 12ms/step - loss: 0.2909 - accuracy: 0.9290 - val_loss: 0.3204 - val_accuracy: 0.9228
Epoch 15/30
57/57 [==============================] - 1s 13ms/step - loss: 0.2907 - accuracy: 0.9303 - val_loss: 0.3206 - val_accuracy: 0.9228
<keras.callbacks.History at 0x7f07ef8b9650>

[85] image_classifier.evaluate(images_normalized, c_train_full)

1563/1563 [==============================] - 12s 8ms/step - loss: 4.3865 - accuracy: 0.9573
[4.386488914489746, 0.9573400020599365]
```

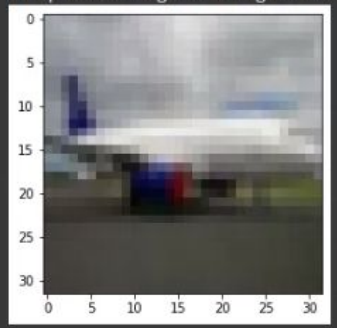
```
i = np.random.randint(10000, 40000)
print(dictionary[int(labels[i])])
plt.imshow(x_noisy[i, :])
```

```
truck
<matplotlib.image.AxesImage at 0x7f20852b13d0>
```




```
i = np.random.randint(10000, 40000)
print(dictionary[int(labels[i])])
plt.imshow(x_noisy[i, :])
```

```
airplane
<matplotlib.image.AxesImage at 0x7f1cdd18ff10>
```




```
i = np.random.randint(10000, 40000)
print(dictionary[int(labels[i])])
plt.imshow(x_noisy[i, :])
```

```
ship
<matplotlib.image.AxesImage at 0x7f1fc026c650>
```