model_building.html

model_evaluation.py

Image Classification {

[Semi-supervised Learning]

<Marvin Limpijankit, Jiachen Liu,
Jiahao Shao, Nichole Zhang, Xile Zhang>

}

Table of ‘Models’ {

01 Baseline Model

<Use Logistic Regression, on noise data only.>

02 Model I

<Use CNN model, fit on noise data only.>

03 Model II

<Improving Model I with label correction process.>

}

Proposed Strategies; {

[Model II]

‘Higher accuracy, but relatively low increase in running cost and storage’

<p % higher accuracy compared to baseline model; 10% higher accuracy compared to model I; lower running cost on test set>

}

```
01 {

   [Model Building]

      < Models are trained using
      the last 49,000 images.
      Saving rest for evaluation >

}
```

Model I 'Layers' {

```
    tf.keras.layers.experimental.preprocessing.Rescaling(1. / 255),
    tf.keras.layers.Conv2D(32, (3, 3), padding='same', activation="relu",
    input_shape=(32, 32, 3)),
    tf.keras.layers.MaxPooling2D((2, 2), strides=2),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation="relu"),
    tf.keras.layers.MaxPooling2D((2, 2), strides=2),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation="relu"),
    tf.keras.layers.Dense(10, activation="softmax")
}
```

Model I ‘Compile & Fit’ {

```
modelI.compile(optimizer=tf.keras.optimizers.Nadam(0.001),
                    loss=tf.keras.losses.CategoricalCrossentropy(),
                    metrics=['accuracy'])

history = modelI.fit(x_train, y_train, batch_size=128, epochs=6,
                     validation_split=0.2,
                     callbacks=EarlyStopping(patience=2))
```

}

Model II ‘Label Correction’ {

```
    # Image branch - Image feature extraction
    Resnet = tf.keras.applications.ResNet50(include_top=False,
                        weights="imagenet", input_tensor=None,
                        input_shape=(32, 32, 3), pooling='max')
    # Fit resnet and scale it down
    img_vec = resnet(img_input)
    img_vec = Dense(1024)(img_vec)
    img_vec = Dense(512)(img_vec)
    img_vec = Dense(256)(img_vec)
```

}

Model II ‘Label Correction’ {

```
    # Noisy label branch
    noisy_l = Dense(10)(noisy_label)
    # Concatenate
    x = Concatenate(axis=-1)([noisy_l, img_vec])
    x = Dense(256, activation='relu')(x)
    out = Dense(10, activation='softmax')(x)
    model = Model([img_input, noisy_label], out)
    # Compile the model
    model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
         metrics=['acc'], optimizer=tf.keras.optimizers.Adam(0.001))
```

}

02 {

[Model Evaluation]

< The first 1,000 data with clean labels are reserved as test set for evaluation only >

}

Baseline Model ‘Confusion Matrix’ {

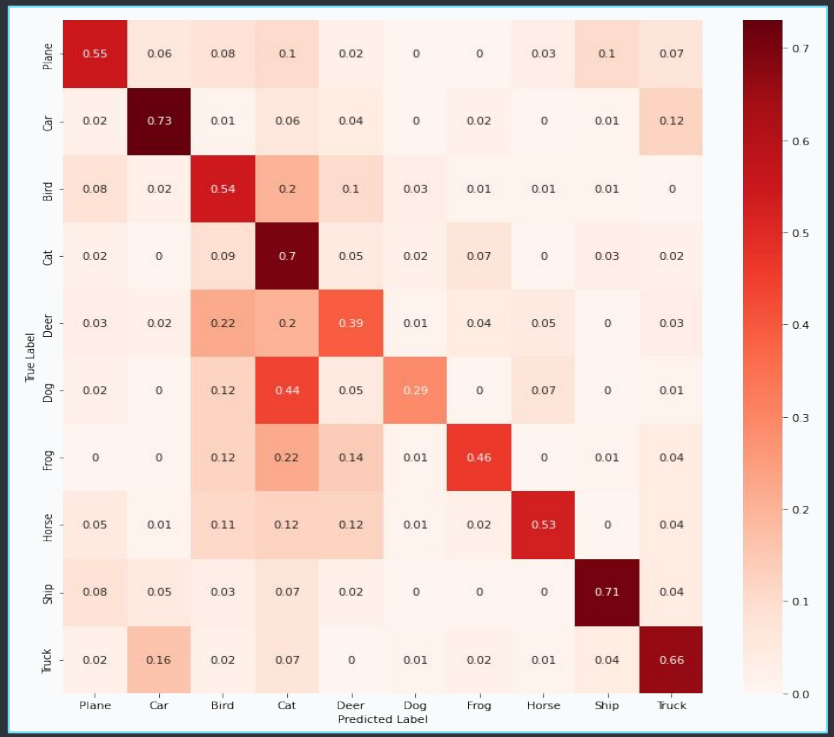
- Chaotic

}

Model I ‘Confusion Matrix’ {

- Clear Concentration on diagonal!
- Except Cat and Dog

}



Model II ‘Confusion Matrix’ {

- Even more concentration achieved!

}

Examples About ‘The Topic’{

Baseline

Training Time

0.184s

Running Time

0.184s

Training/Model Storage

--MB

Accuracy

25%

Model I

Training Time

138.522s

Running Time

20.943s

Training/Model Storage

6.7MB

Accuracy

56%

Model II

Training Time

1,073.510s

Running Time

21.086s

Storage Including Label Correction

13.6MB

Model Storage

6.7MB

Accuracy

66%

}

Thanks;) {

‘Do you have any questions?’

GitHub:
https://github.com/TZstatsADS/spring-2022-prj3-group11

}