Project 3: Weakly supervised learning
Label noise and correction

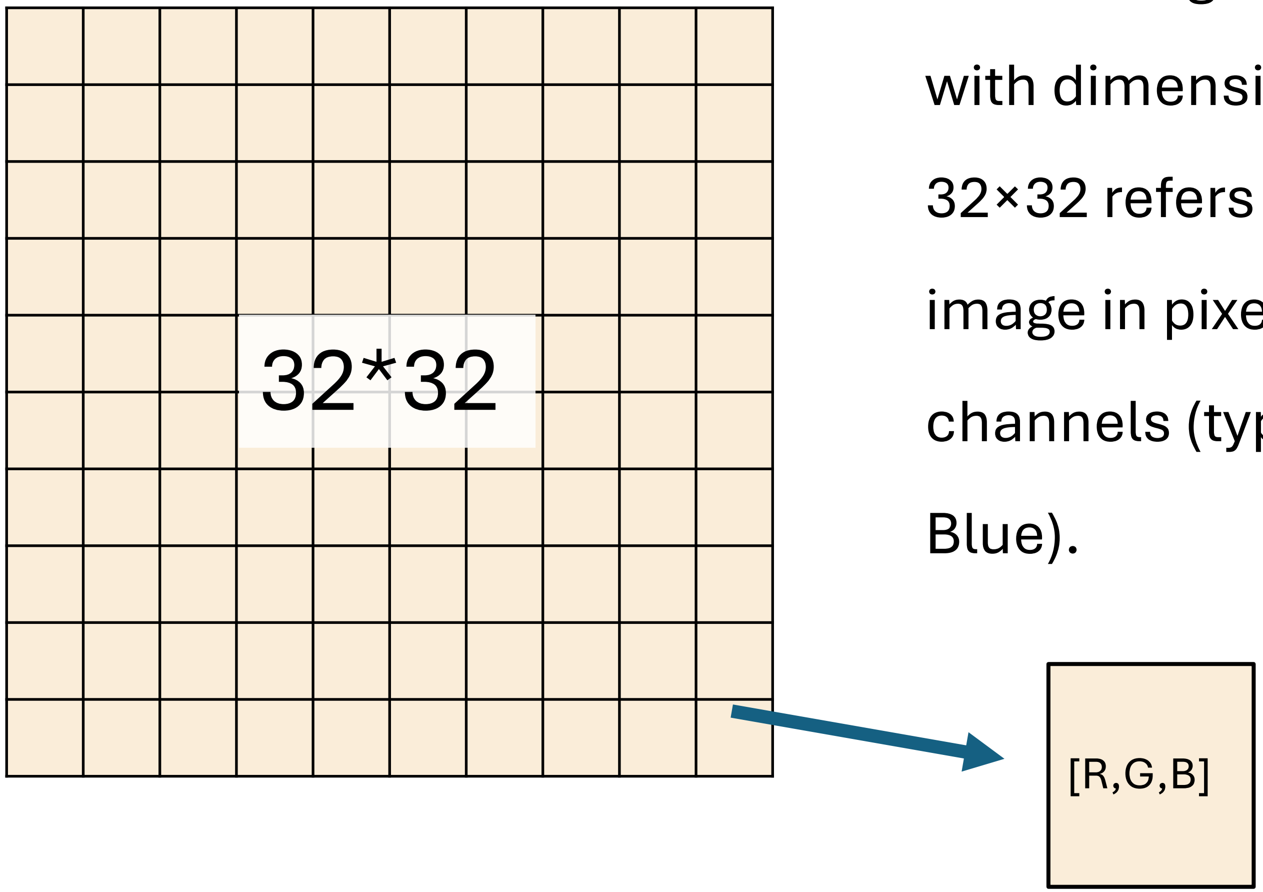
Wenjun Yang, Licheng Wu, Forain Zhang, Yiwei Jiang

Summary

In our project, we created models to classify 50k images into 10 classes. Our dataset comprises labels, which contain some inaccuracies, and an additional set of 10k verified, error-free labels.

- Model 1: This model is based on a Convolutional Neural Network (CNN) approach. For training purposes, we use the 49k noisy labels treating them as clean data.
- Model 2: The second model integrate a label correction mechanism designed specifically to address the inaccuracies within the noisy labels, thereby enhancing the reliability of the training data Zhou,2017).

Load Dataset-Image



Converting image information into a tensor with dimensions 32×32×3. The dimension 32×32 refers to the width and height of the image in pixels, and 3 refers to the color channels (typically RGB: Red, Green, Blue).

Dataset - Labels

Labels	Class
0	plane
1	car
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

Let Y represent the noisy labels.

Let V represent the clean labels.

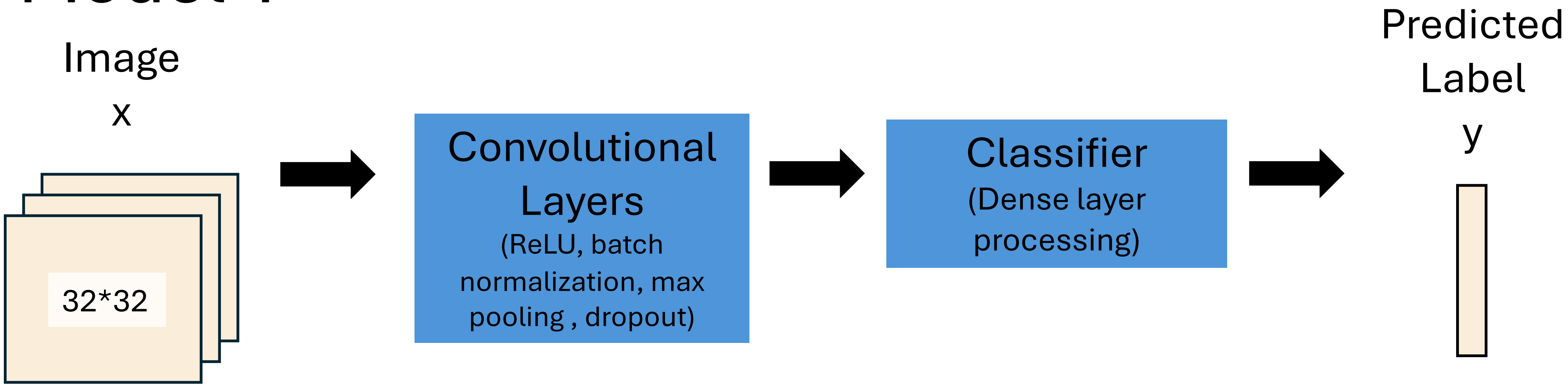
The tensor in dimension 10×1

represent the class.

car

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Model 1



Steps:

1. Start with a sequential model to layer operations linearly.
2. Apply convolutional layers.
3. Flatten convolutional output to a 1D vector for dense layer processing.
4. Add a dense layer with many neurons for complex pattern recognition.
5. Compile the model focusing on accuracy, using Adam optimizer and categorical loss.

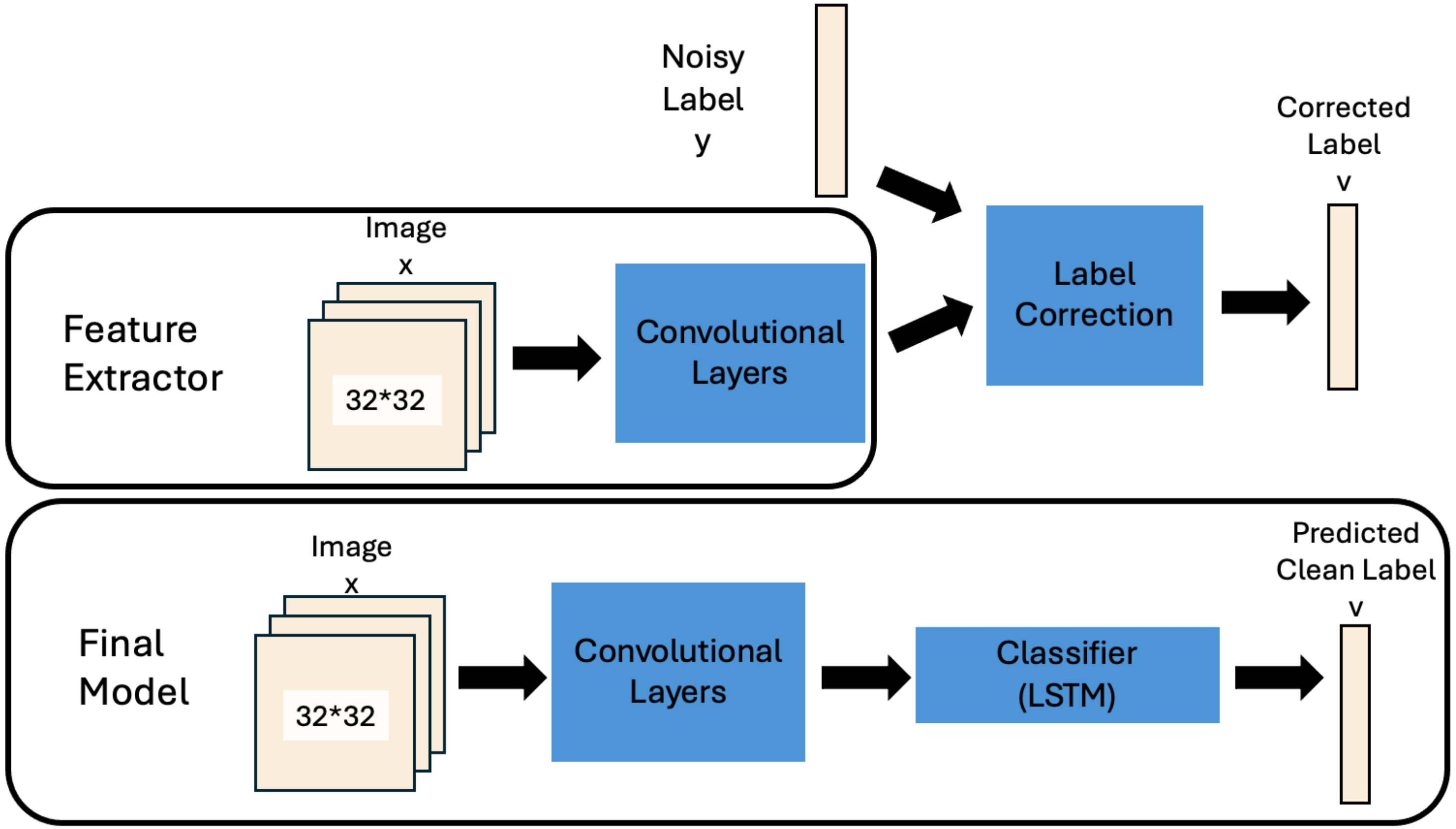
Model 1 -Performance

Model	Accuracy	Runtime
Model 1	0.4720	821.61 seconds

```
Epoch 1/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 84s 65ms/step - accuracy: 0.1206 - loss: 2.7503 - val_accuracy: 0.1388 - val_loss: 3.0546
Epoch 2/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 81s 65ms/step - accuracy: 0.1519 - loss: 2.4058 - val_accuracy: 0.2125 - val_loss: 2.2662
Epoch 3/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 81s 65ms/step - accuracy: 0.1647 - loss: 2.3489 - val_accuracy: 0.3250 - val_loss: 1.9991
Epoch 4/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 81s 65ms/step - accuracy: 0.1652 - loss: 2.3106 - val_accuracy: 0.2945 - val_loss: 2.0657
Epoch 5/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 82s 65ms/step - accuracy: 0.1689 - loss: 2.2906 - val_accuracy: 0.3878 - val_loss: 1.9370
Epoch 6/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 82s 65ms/step - accuracy: 0.1807 - loss: 2.2679 - val_accuracy: 0.3882 - val_loss: 1.9419
Epoch 7/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 82s 65ms/step - accuracy: 0.2014 - loss: 2.2463 - val_accuracy: 0.4310 - val_loss: 1.8101
Epoch 8/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 82s 65ms/step - accuracy: 0.2038 - loss: 2.2389 - val_accuracy: 0.3736 - val_loss: 1.9128
Epoch 9/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 82s 65ms/step - accuracy: 0.2110 - loss: 2.2282 - val_accuracy: 0.4324 - val_loss: 1.8958
Epoch 10/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 82s 65ms/step - accuracy: 0.2188 - loss: 2.2166 - val_accuracy: 0.4720 - val_loss: 1.8525
Training accuracy:  0.21995000541210175
Validation accuracy:  0.47200000286102295
313/313 - 3s - 10ms/step - accuracy: 0.4720 - loss: 1.8525

Test accuracy: 0.47200000286102295
Runtime of the first part: 821.6063859462738 seconds
```

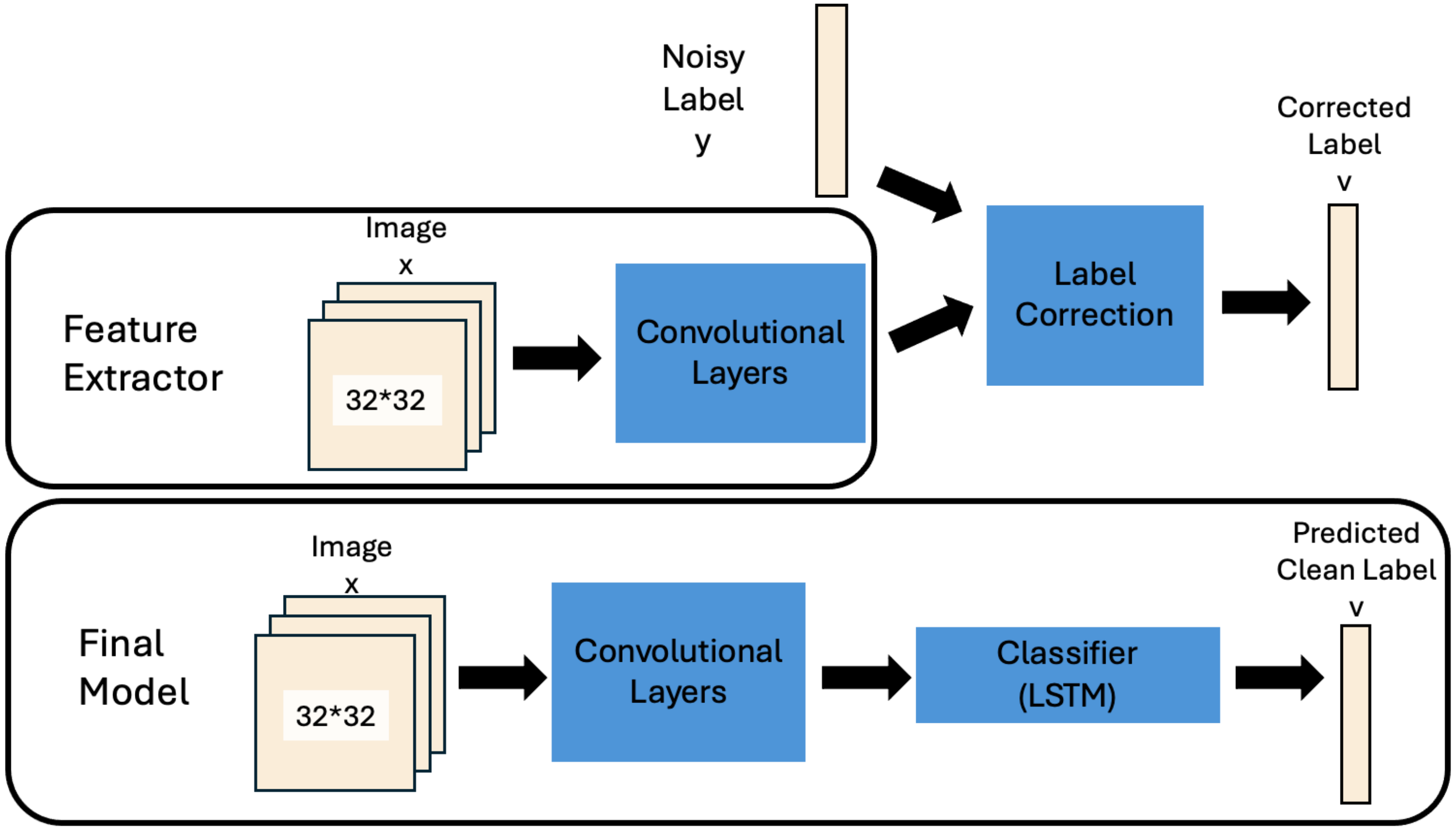
Model 2



Part one:

1. Implement `ReduceLROnPlateau` callback to reduce learning rate when validation loss stops improving and `ModelCheckpoint` to save the model with the best validation accuracy.
2. Assemble the main model, integrating feature extraction with processing of noisy label inputs.
3. Compile the model, specifying the optimizer, loss function, and metrics for performance evaluation.
4. Employ K-Fold cross-validation to train the model systematically on different data subsets.
5. Fit the model using both clean images and noisy labels, applying specified callbacks.

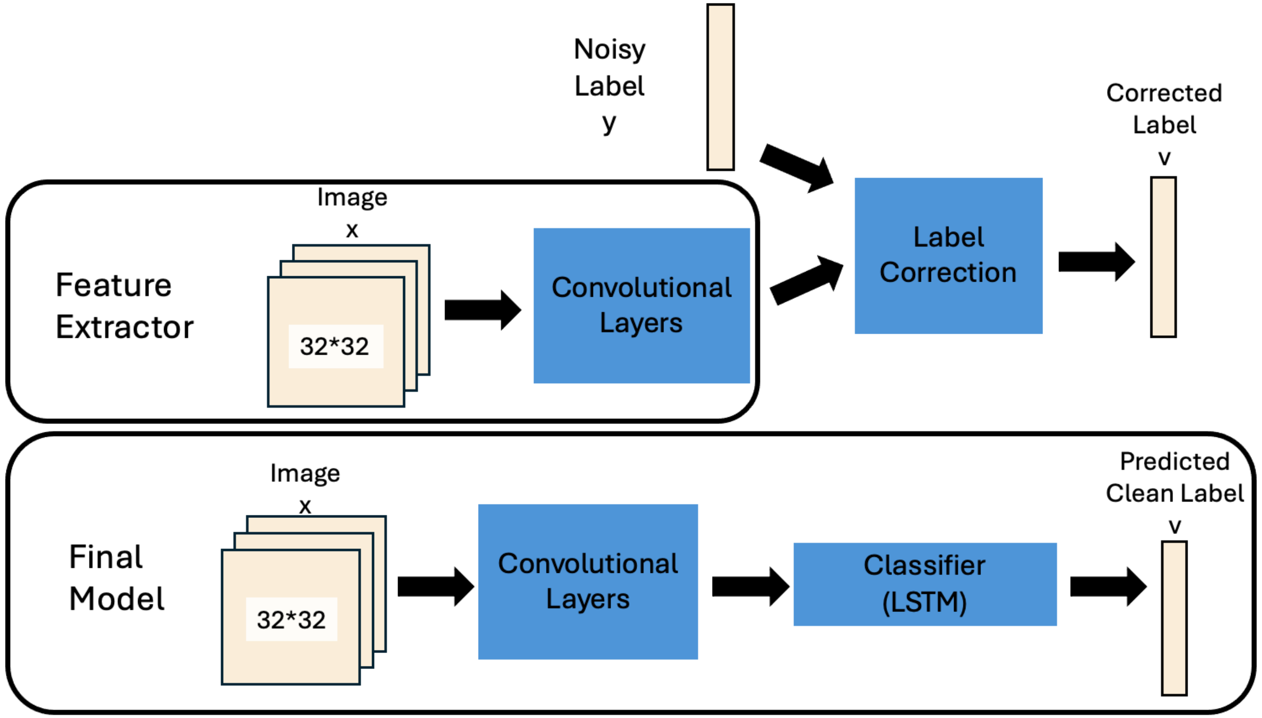
Model 2



Part two:

1. Predict probabilities for noisy images using both image features and noisy labels.
2. Convert prediction probabilities to class labels by selecting the class with highest probability.
3. Apply a confidence threshold of 0.9 to filter predictions with high certainty.
4. Identify indices of predictions exceeding the confidence threshold for reliable pseudo-label generation.
5. Generate pseudo labels for high-confidence predictions to use in further model training/enhancement.

Model 2



Part three:

1. Apply convolutional layers with 24 filters each for initial feature extraction, max pooling, and more convolutional layers with increased filters(Dharmaraj, 2022)..
2. Flatten and reshape the output from convolutional layers to prepare for LSTM processing.
3. Integrate an LSTM to analyze image features, dense layers for high-level reasoning, a dropout layer to prevent overfitting, and output layer with softmax activation to classify images.
4. Compile the model with Adam optimizer and sparse categorical crossentropy for classification.
5. Train the model using combined clean and pseudo-labeled images for robust learning with K-Fold cross-validation to ensure model generalizes well across different data splits.
6. Utilize callbacks for dynamic learning rate adjustments and to save the best-performing model.

Model 2 -Performance

Model 2	Accuracy	Runtime
Model without k-fold and LSTM layers	0.8424	73.98 seconds
Model with k-fold and LSTM layers	0.9184	382.77 seconds

Comparison

Model	Accuracy
Baseline	0.24
Model 1	0.4720
Model without k-fold and LSTM layers	0.8424
Model with k-fold and LSTM layers	0.9184

Predictive Performance (1k test)

- The weight avg is 0.6148.

```
[61]: classification_report(clean_labels[:1000] ,model2_label,zero_division=0, output_dict=True)

[61]: {'0': {'precision': 0.7088607594936709,
  'recall': 0.5490196078431373,
  'f1-score': 0.6187845303867404,
  'support': 102.0},
 '1': {'precision': 0.8043478260869565,
  'recall': 0.6607142857142857,
  'f1-score': 0.7254901960784315,
  'support': 112.0},
 '2': {'precision': 0.55,
  'recall': 0.4444444444444444,
  'f1-score': 0.4916201117318436,
  'support': 99.0},
 '3': {'precision': 0.3815789473684211,
  'recall': 0.31521739130434784,
  'f1-score': 0.34523809523809523,
  'support': 92.0},
 '4': {'precision': 0.5436893203883495,
  'recall': 0.5656565656565656,
  'f1-score': 0.5544554455445545,
  'support': 99.0},
 '5': {'precision': 0.4935064935064935,
  'recall': 0.4470588235294118,
  'f1-score': 0.4691358024691358,
  'support': 85.0},
 '6': {'precision': 0.5923076923076923,
  'recall': 0.719626168224299,
  'f1-score': 0.649789029535865,
  'support': 107.0},
 '7': {'precision': 0.693069306930693,
  'recall': 0.6862745098039216,
  'f1-score': 0.6896551724137931,
  'support': 102.0},
 '8': {'precision': 0.7073170731707317,
  'recall': 0.8787878787878788,
  'f1-score': 0.7837837837837839,
  'support': 99.0},
 '9': {'precision': 0.6115107913669064,
  'recall': 0.8252427184466019,
  'f1-score': 0.7024793388429752,
  'support': 103.0},
 'accuracy': 0.616,
 'macro avg': {'precision': 0.6086188210619914,
  'recall': 0.6092042393754894,
  'f1-score': 0.6030431506025218,
  'support': 1000.0},
 'weighted avg': {'precision': 0.6147993059530343,
  'recall': 0.616,
  'f1-score': 0.609393072444342,
  'support': 1000.0}}
```

Reference

- Inoue, N., Simo-Serra, E., Yamasaki, T., & Ishikawa, H. (2017). Multi-label fashion image classification with minimal human supervision. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. https://doi.org/10.1109/iccvw.2017.265
- Zhou, Z.-H. (2017). A brief introduction to weakly supervised learning. *National Science Review*, *5*(1), 44–53. https://doi.org/10.1093/nsr/nwx106
- GfG. (2023, January 4). *Python opencv: Cv2.cvtcolor() method*. GeeksforGeeks. https://www.geeksforgeeks.org/python-opencv-cv2-cvtcolor-method/
- Dharmaraj. (2022, June 1). *Convolutional Neural Networks (CNN) - architectures explained*. Medium. https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243