**Files**

**A. Preprocessing**

**1. Introduction and EDA**

In this file, we introduce the problem statement and the files we are given.

Then, we perform exploratory data analysis on the train.csv file to examine (a) the csv file as a whole and (b) the features individually.

**2. Images to images.csv**

We convert the images in the train\_image directory to a single csv file where the images are converted to a row of 12288 (64x64x3) RGB values (0 -255)

**3. Check images.csv**

In this file, we check by index that the image\_file.jpg in the train dataframe matches the image\_file.jpg in the images.csv.

We convert some samples in the images.csv back to images to make sure that the conversion was done correctly.

**4. Title Translation**

We noticed that the title text of the train.csv and test.csv files are a mix of Indonesian and English. We translate the title to English, and export train\_translate.csv and test\_translate.csv.

**5. Sampling**

For model testing purposes, we randomly select 500 observations from the images.csv file and the train.csv file. These samples have matching indexes. A simple check on matching labels is performed.

The sample csv files are exported.

**6. Train Validation Split**

We noticed that every unique label has at least one duplicate.

The problem with using keras train\_test\_split function: there is a large number of unique labels. If the given test label is not in the training data, the model will never predict accurately on the label it has never seen before. The model only classifies the observation with a label given in the training labels.

We took 20% samples out of the duplicated rows. In this case, all the unique labels remain in the training data.

**B. Model Testing**

**Baseline Model 1: Standalone Keras-CNN Model (Images Only)**

We trained a Keras-CNN model using the subsetted training data from Train-Validation split. In 10 splits, we achieved a training accuracy of 0.0013 and a validation accuracy of 0.0017.

**Grid Search: Standalone Keras-CNN Model (Images Only)**

We experimented in finding the best parameters for the standalone Keras-CNN model. This test model is built on a sample dataset of 500 images.

This process took 3 hours. The results did not deviate from our initial parameters.

Due to time considerations and resource concerns, we do not think it’s worth it to run grid search on our final model.

**Baseline Model 2: Standalone TFIDF RNN Model (Title Only)**

**TFIDF-CNN Model**

We combined TFIDF embedding and CNN layers. In 10 epochs, we achieved a training accuracy of 0.68 and a testing accuracy of 0.41.

Our final model is the TFIDF-CNN model with an epoch of 25. The training accuracy is 0.98 with a testing accuracy of 0.51.

**BERT - CNN Model**

We experimented with combiningTFBERT model with image CNN layers. In this model, title texts are converted to 128 tokens by BERT embedding.

We achieved a training accuracy score of 0.05 after 10 epochs. It takes 15 minutes per epoch.

The performance may be low because of the limited number of tokens from title embedding. However, 128 tokens is the best we can do given 40GB of GPU RAM.

This model does not seem practical moving forward. More examination and experimentation is needed to make this model work.

**GPT - CNN Model**

We experimented with combiningTFGPT2 model with image CNN layers. In this model, title texts are converted to 128 tokens by GPT2 embedding.

We achieved a training accuracy score of 0.95 after 10 epochs. It takes 2 minutes per epoch. However the testing accuracy is 0.000216.

The performance may be low because of the limited number of tokens from title embedding. However, 128 tokens is the best we can do given 40GB of GPU RAM.

This model does not seem practical moving forward. More examination and experimentation is needed to make this model work.

**C. Application Building**

**1. Model Testing**

We import the model and use the final model to make label predictions on the 3 test.csv observations.

**2. Final Application**

We build a final model that takes a dataframe (test.csv) as input, and provides a table of matching post\_ids as output.

**Next Steps**

* Image resolution (https://pubs.rsna.org/doi/full/10.1148/ryai.2019190015)
* Model finetuning
* Only 1 label predicted. Same image, 2 labels (2%)