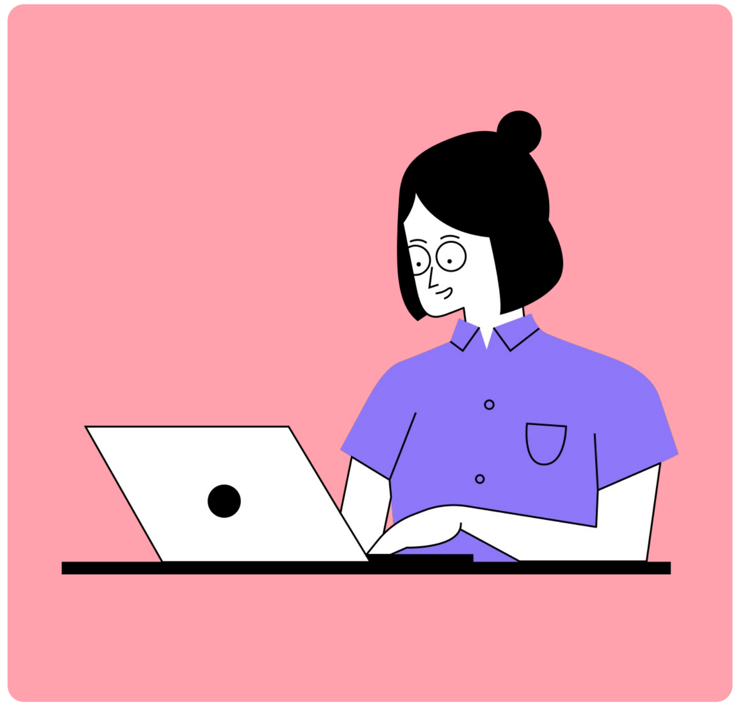
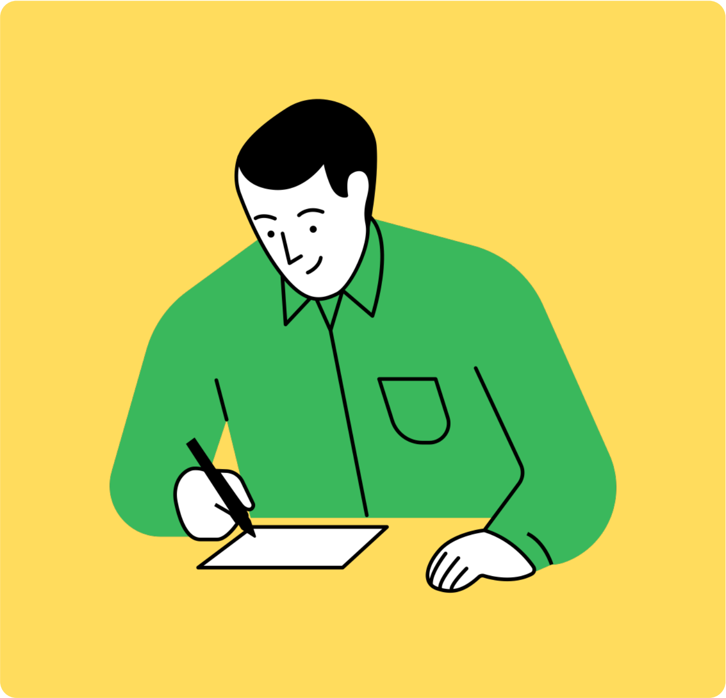
Document Image Classification



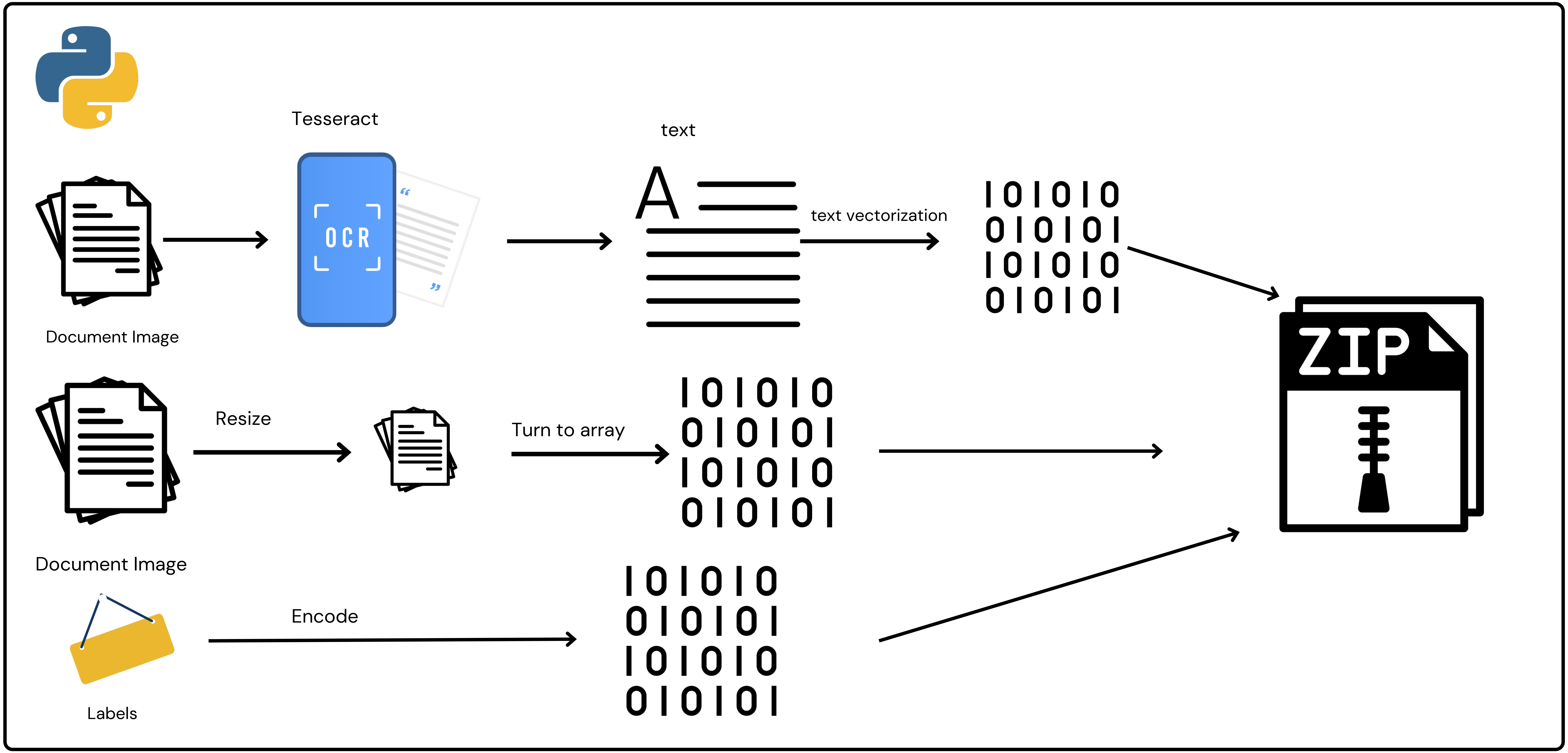
Data preparation



Model Building

Model Evaluation

Data Preparation

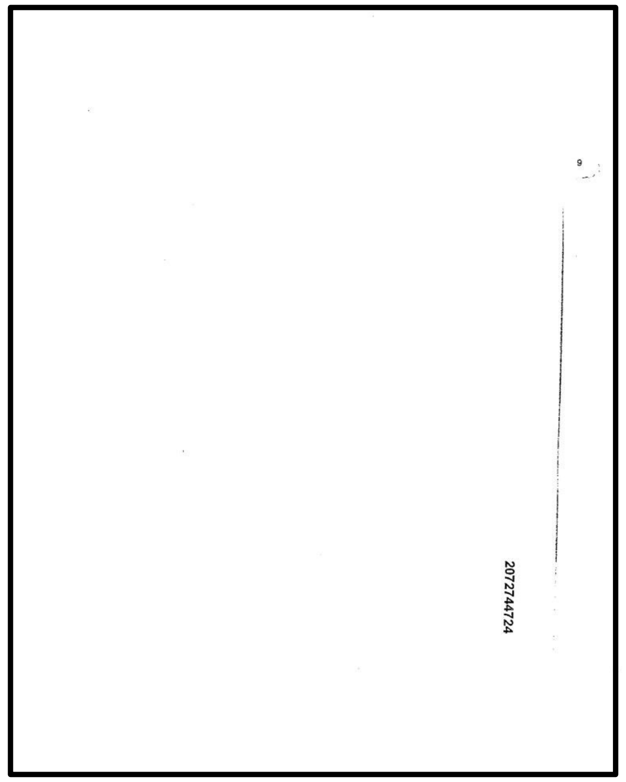


Data Preparation

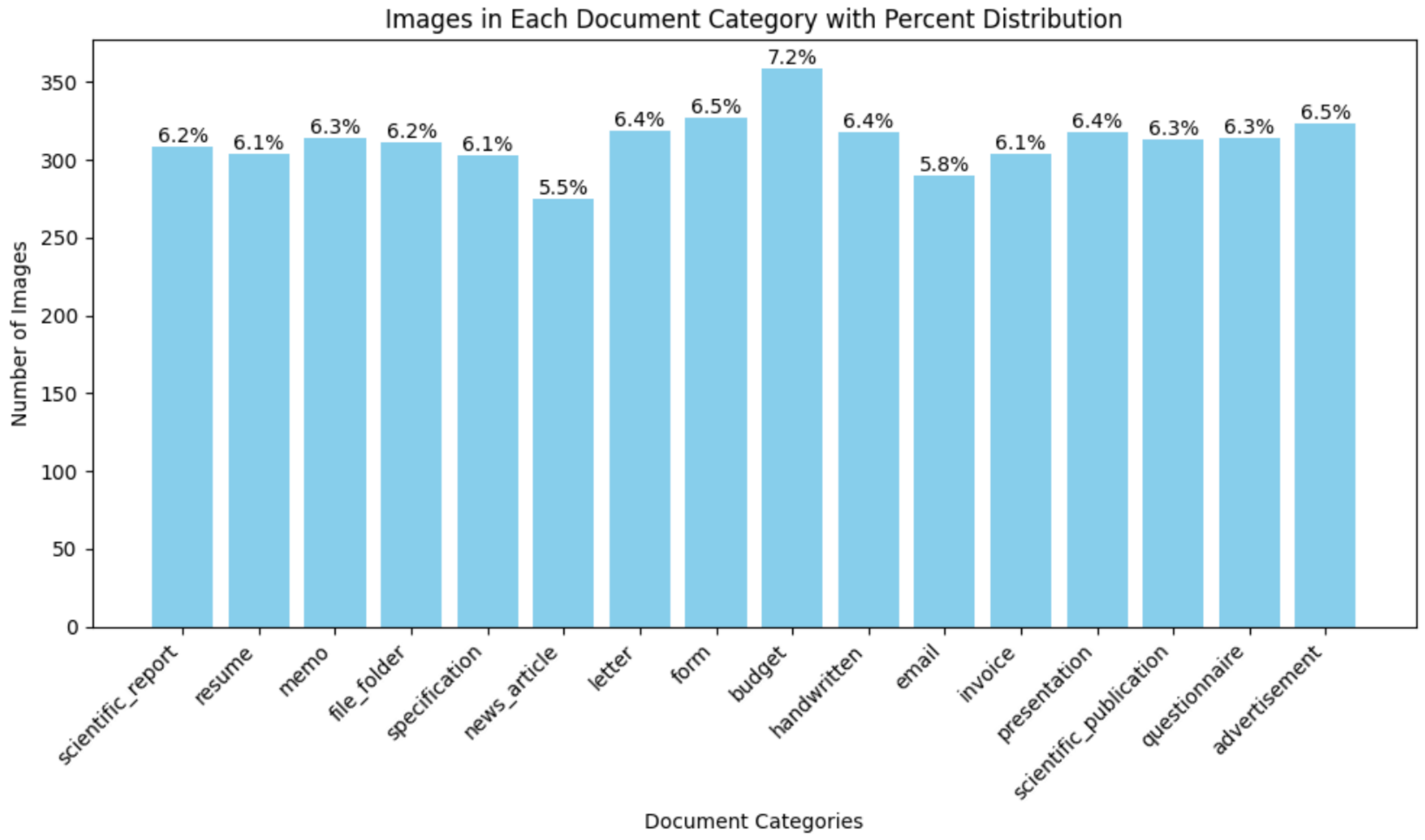
Train 70%	Valid15%	Test15%

Lead to mistranslation

Rotated page



Almost blank page



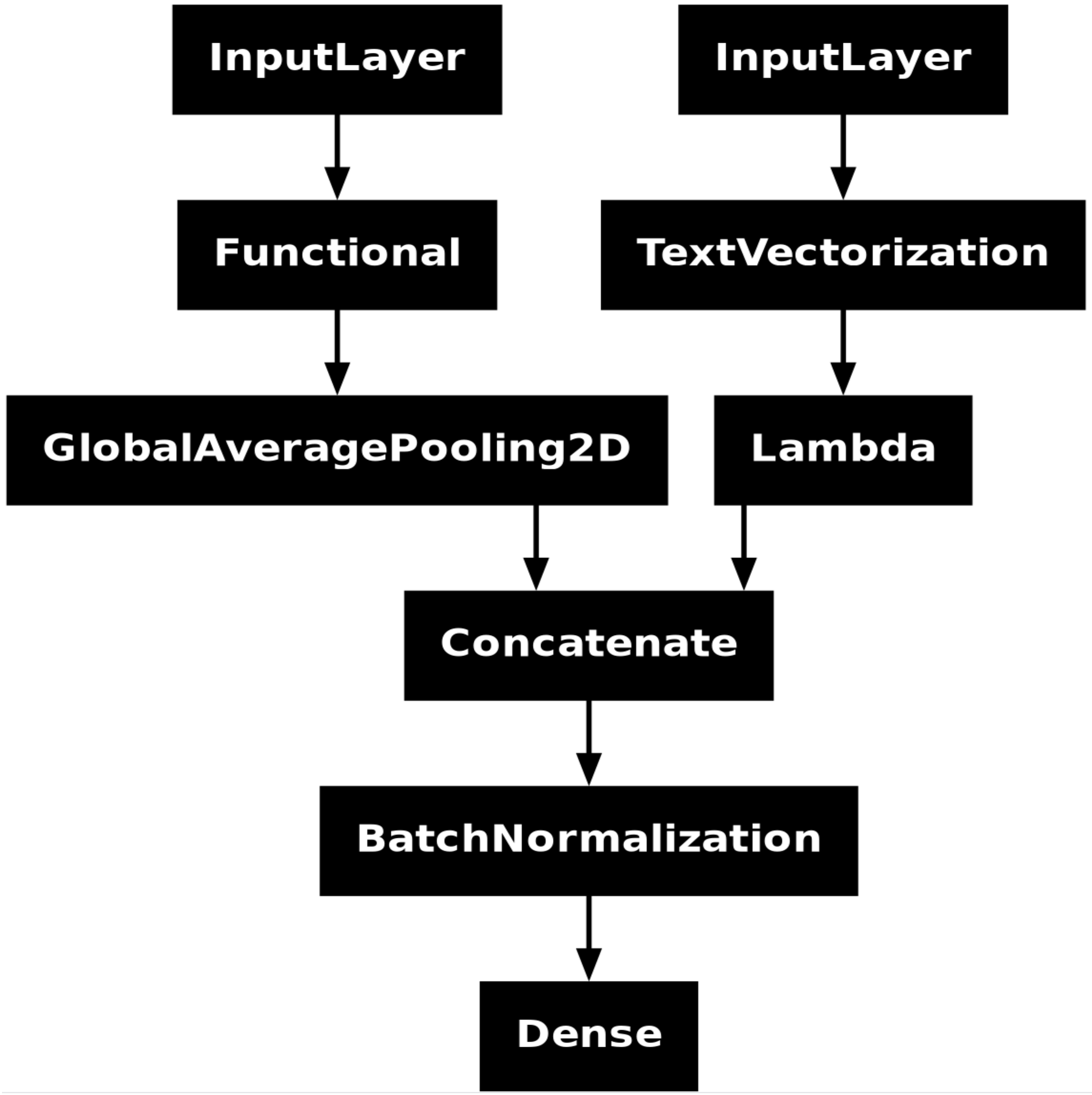
For the rotated image, Possible solution

- Image classification
- Rotation and evaluate with OCR

For the blank page, Possible solution

- Drop page filter by len(text)< setting value

Model training



VGG19 pretrained model extract image text feature by after the convolutional base of VGG19 helps to extract compact and informative representations of input images.

Text model are text vectorization which convert input text then normalizes the data

Concatenate both feature and pass through normalization layer

All normalized data get into Dense layer with softmax activation to get the predicted class

Model training

- Batch size =32 Smaller batch help from overfitting but slow down the convergence
- Vocabulary size=50,000 to increase capacity of token after tokenize the text
- Set train and valid data are a number of sample during training
- Step per epoch = a number of batch which model process in one epoch

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 512, 512, 3)	0	-
input_layer_2 (InputLayer)	(None, 1)	0	-
vgg19 (Functional)	(None, 16, 16, 512)	20,024,384	input_layer_1[0]…
text_vectorization (TextVectorization)	(None, 50000)	0	input_layer_2[0]…
global_average_poo… (GlobalAveragePool…	(None, 512)	0	vgg19[0][0]
lambda (Lambda)	(None, 50000)	0	text_vectorizati…
concatenate (Concatenate)	(None, 50512)	0	global_average_p… lambda[0][0]
batch_normalization (BatchNormalizatio…	(None, 50512)	202,048	concatenate[0][0]
dense (Dense)	(None, 16)	808,208	batch_normalizat…

Total params: 21,034,640 (80.24 MB)
Trainable params: 20,933,616 (79.86 MB)
Non-trainable params: 101,024 (394.62 KB)

Number of Trainable parameter

Loss Function: Sparse categorical Entropy for categorical index and for index 1-16

Optimizer: Adam(Adaptive Moment Estimation): Tend to coverage faster and not much sensitive

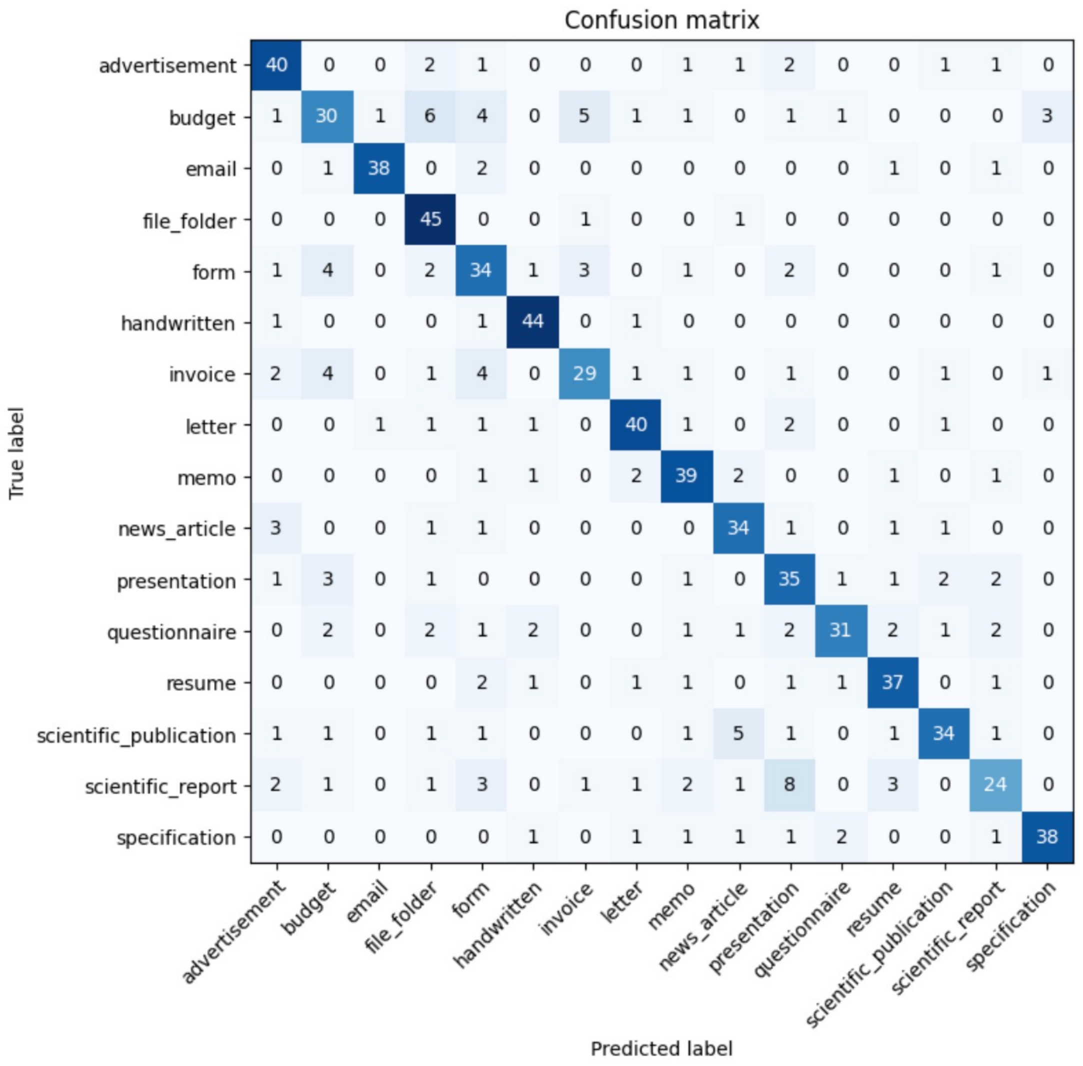
This case have multiple class so we use multi classification

Model Evaluation

Evaluation Metric: Accuracy, Since the data is seem to be balance so accuracy can optimize overall metrics and easy to be explainable
Alternative: F1 Score can choose which balance between precision and recall

	precision	recall	f1-score	support
advertisement	0.7692	0.8163	0.7921	49
budget	0.6522	0.5556	0.6000	54
email	0.9500	0.8837	0.9157	43
file_folder	0.7143	0.9574	0.8182	47
form	0.6071	0.6939	0.6476	49
handwritten	0.8627	0.9362	0.8980	47
invoice	0.7436	0.6444	0.6905	45
letter	0.8333	0.8333	0.8333	48
memo	0.7647	0.8298	0.7959	47
news_article	0.7391	0.8095	0.7727	42
presentation	0.6140	0.7447	0.6731	47
questionnaire	0.8611	0.6596	0.7470	47
resume	0.7872	0.8222	0.8043	45
scientific_publication	0.8293	0.7234	0.7727	47
scientific_report	0.6857	0.5106	0.5854	47
specification	0.9048	0.8261	0.8636	46
accuracy			0.7627	750
macro avg	0.7699	0.7654	0.7631	750
weighted avg	0.7675	0.7627	0.7605	750

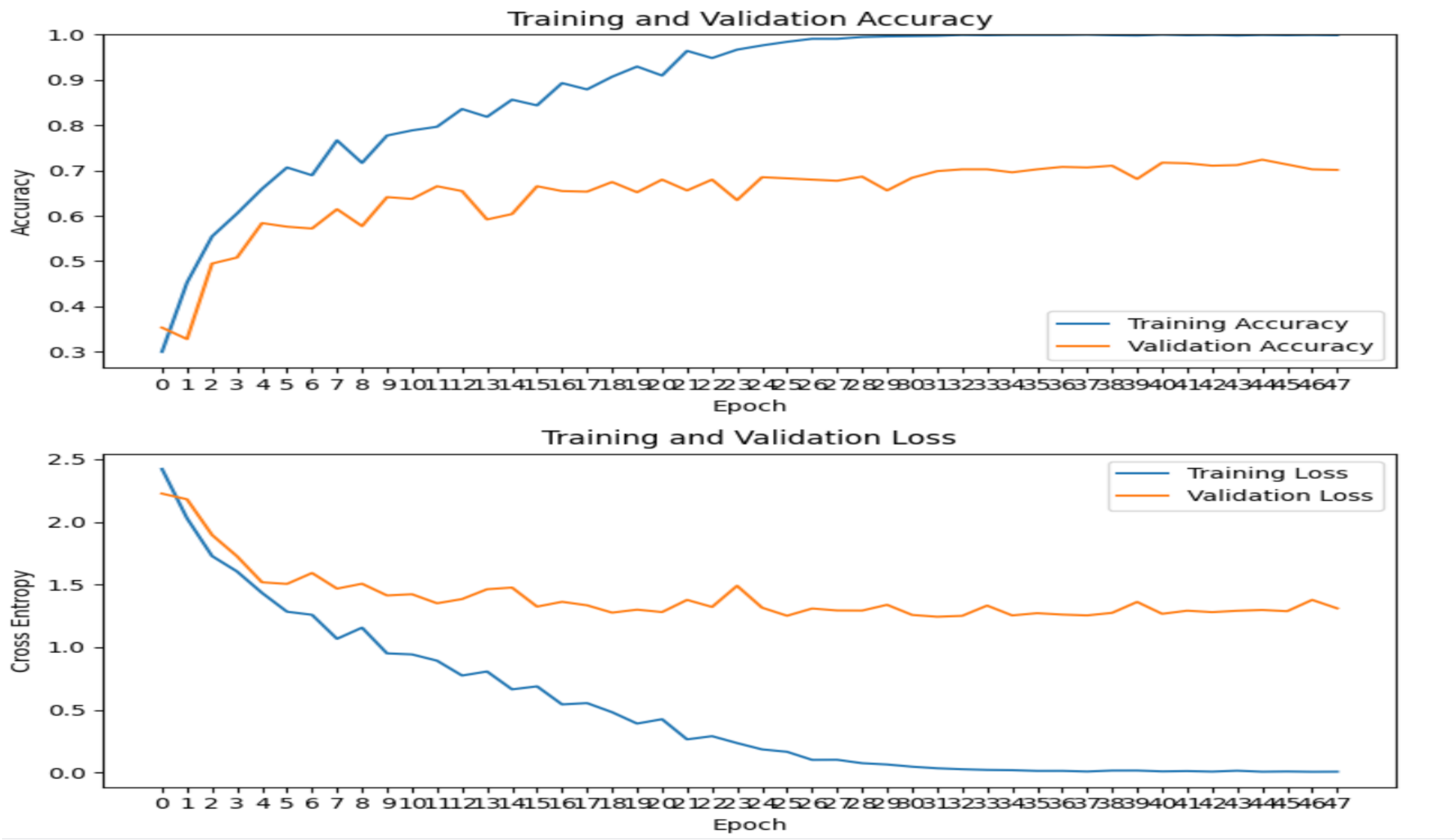
Classification report



Confusion matrix

Model Evaluation

Since the model is overfitted
There are a large gap between training and validation accuracy and loss. So the model might not work well with unseen data



Solution

- Make model Generalize
    - Regularization(L1,L2,Droput,Batchnormalization)
- Data Augmentation
- Simplify model to be less complex
- Hyperparameter tuning

Note: There are trade off between best performance and overfitting

Suggestion

- Improvement for the performance
  - Hyperparameter tuning
  - Try other pretrained model and embedding method
  - Optimize OCR by rotate image (Data Cleansing)
  - Build custom Sequence model for text model (Eg:LSTM)
  - LLM integration
  - Increase training size
- Improve runtime
  - Accelerate by GPU/TPU
  - Find other OCR tool which can run on GPU