(a) (25pt) Do your "best" to get good results. Ideal results distinguish all these results 100% perfectly, work fast and contain only 'few parameters. But this is probably unachievable goal.

We spent most of our effort trying different types of models to get the greatest validation accuracy with the fewest amount of parameters. Adding layers, altering kernel sizes, and modifying the epoch size according to the overfitting/underfitting findings took endless hours (about 30 hours in total).

When having to deal with two languages (EN, ZN & EN, DA), we were able to regularly achieve high validation accuracies, but when we added more languages to the model, it became much more difficult. We decided to spend more time studying the dataset, the code, and the parameters we were tweaking after spending a significant amount of time testing models that were only able to attain approximately 50%-60% accuracy.

As a result, we began to examine the pictures in the training set of various languages in order to better comprehend the reasoning behind the misclassification. We also spent time on the internet looking for ways to improve our outcomes by visiting sites that focused on convolutional neural networks and the hyperparameters that are essential for a successful model. We came across an article online that ultimately helped us achieve the high consistent accuracies that we got for most of the models.

During the testing portion of this problem set (Suggested Tasks), we had decided to take a random sample of 1000-5000

(b) (30pt) Explain what did you try and what did you get.
i. List here different models, layer structures, image pre-processing that you tried.

Models:

- All models were made linearly using Sequential()
- Used several different kernel sizes, strides, and even activation functions
    - relu, sigmoid | (3x3), (2x2) kernel | (2x2), (1x1) strides

Layer Structures:

- Layer types used:
    - Conv2D, Dense, Flatten, MaxPooling2D
- Conv layer input sizes tried: 16, 32, 64, 128
- Density layer sizes tried: 100, 128, 200, 256, 512, 1024
- Padding
    - Experimented with 'same' and 'valid' layer padding in order to see how zero-padding would influence training accuracy

Pre-processing:

- Cropping Images
- Image Target Size (height & width)
    - (25x25), (35x35), (50x50)
- Testing 2 languages, 3 (mix of combinations), and finally all at once
    - EN & ZN - 96%

- EN & DA - 95.1%
- EN, ZN, TH - 75.3%
- EN, ZN, TH, DA, RU - 85%

Final Model Structure & Hyperparameters:

- Input Details:
  - Kernel (3x3)
  - Image target size (50x50)
  - Strides (2x2)
  - Padding - 'valid' (or none)
- Layers:
  - Size 32 input 2D convolutional layer
  - (2x2) MaxPooling2D layer
  - Density layer 200 (relu activation function)
  - Density layer nCategories (5) (softmax activation function)
- Loss:
  - Categorical cross entropy

The above-described statistics and details encapsulate all of our various methods and attempts at creating a high-accuracy model. Overall, we arrived at the conclusion that simplicity and understanding of the model are more valuable than trial and error.

Our first approach was to play with layer input sizes and density layer sizes, for which we devised a function to iterate through convolutional layer sizes [16, 32, 64, 128] and density layer sizes [128, 256, 512, 1024] to test out every possible combination. This produced a wide range of models to choose from, however, it was ultimately not the smartest approach as simply varying layer sizes and amounts do not create the best model.

After reading and researching online, we found that number-based image classification models run 96%+ accuracy with only one convolutional layer, which led us to use a single Conv2D layer before the two density layers and tune our hyperparameters and function attributes accordingly.

iii. Explain also the technical side. Was your computer good enough? Did you use some other kind of computing resources? Did you have to scale down the dataset? Were you able to run enough epochs?

As a group of 2, one of our computers is a Macbook Pro 2020, using the relatively new M1 chip and Surface 4 laptop. The Macbook could not install TensorFlow, so we had decided to just use the Surface 4 laptop, with a processor base clock of 1.20GHz and a max clock of 3.70GHz (4 cores, 8 threads). Although we only had 1 computer to work with, we still had decided to not scale down the dataset because we wanted to yield the best results we could possibly get. In terms of the number of epochs, we didn't have any problems with it and were able to run as many epochs as we needed.

(c) (20pt) Present your “final results”, your model that you consider the best (this should be the code your submit), and its main hyperparameters and performance indicators. How many epochs did you use? How long time did it take? Show your confusion matrix and compute accuracy.

For our final results, we got a 92% validation accuracy using all the training data. However, it took about 17 minutes to run the entire model. When we used a random sample of 5000 for the training data, we got roughly 85% accuracy but it only took 1-2 minutes to run. We initially did not prioritize the number of hyperparameters there were for our models but when we started to get high consistent accuracies, we started to adjust the layer sizes and densities to slightly lower the parameters. Our goal was to lower the parameters to below 1 million, starting from a maximum of about 10 million. Based on running our final model using all the training data, we were able to achieve our goal with a total of 924, 185 parameters with 0 non-trainable parameters.

Final Results Below:

```
=================================================================
Total params: 924,185
Trainable params: 924,185
Non-trainable params: 0
_________________________________________________________________
Found 31869 validated image filenames belonging to 5 classes.
Epoch 1/8
996/996 [==============================] - 258s 258ms/step - loss: 1.1380 - accuracy: 0.5006
Epoch 2/8
996/996 [==============================] - 100s 101ms/step - loss: 0.7705 - accuracy: 0.6980
Epoch 3/8
996/996 [==============================] - 101s 102ms/step - loss: 0.6056 - accuracy: 0.7404
Epoch 4/8
996/996 [==============================] - 84s 85ms/step - loss: 0.5329 - accuracy: 0.7551
Epoch 5/8
996/996 [==============================] - 91s 92ms/step - loss: 0.4923 - accuracy: 0.7671
Epoch 6/8
996/996 [==============================] - 83s 83ms/step - loss: 0.4116 - accuracy: 0.8534
Epoch 7/8
996/996 [==============================] - 118s 118ms/step - loss: 0.2385 - accuracy: 0.9507
Epoch 8/8
996/996 [==============================] - 108s 109ms/step - loss: 0.1534 - accuracy: 0.9600
7967 validation images
7967 validation files read from languages/validation
Found 7967 validated image filenames.
 --- Predicting on validation data ---
249/249 [==============================] - 73s 296ms/step
Predicted probability array shape: (7967, 5)
Example:
 [[0.00403608 0.00422058 0.00462585 0.97876215 0.00835534]
 [0.48285195 0.2473492  0.24763617 0.01227709 0.00988564]
 [0.36016715 0.6023716  0.02894862 0.00440976 0.00410293]
 [0.2870684  0.57924753 0.12169458 0.00670174 0.00528778]
 [0.95299673 0.02215645 0.01431635 0.00469623 0.00583424]]
                            filename category  predicted
0  aakjaer-samlede-verker-1_DA-aaa.jpg       DA          3
1  aakjaer-samlede-verker-1_DA-aad.jpg       DA          0
2  aakjaer-samlede-verker-1_DA-aai.jpg       DA          1
3  aakjaer-samlede-verker-1_DA-aan.jpg       DA          1
4  aakjaer-samlede-verker-1_DA-aau.jpg       DA          0
confusion matrix (validation)
predicted   DA    EN    RU    TH    ZN
category
DA         833   174     6     5     1
EN         193  1828    26     1     2
RU         111    78  1500     1     3
TH          11     5     2  1875     6
ZN          16     1     1     4  1284
```