Name: Hannes Bischoff
Academic year: 2020/2021
Course: intelligent Systems

Hand gesture detector on android phone with pre trained TensorFlow Light Model

MOTIVATION

The original idea behind this project was to develop an android app which uses the camera to detect different hand gestures. This could assist people who don't understand fingerspelling or the hand alphabet to 'read' or understand what a person shows in sign language. This is a very complicated and complex process. I was thinking about how to reduce complexity to create an achievable and realistic project for the Intelligent Systems course. This project should more or less be a proof of concept that it is possible to detect different hand gestures on an android phone by using the convolutional neural network (cnn) with a pre trained TensorFlow Light Model for image classifying. Furthermore this small report will highlight the limitations and challenges I experienced in this process.

TECHNOLOGY

The artificial intelligence part will be covered by the convolutional neural network. I decided to use TensorFlow because it was used in the exercises during the semester so I already used it and had experience in it and because the light version of TensorFlow models could be run in an android app, which makes it perfect for mobile use in the described use case.

Especially for the transfer learning for image classification I used TensorFlow Lite model maker library with Keras as base layer of the model.

GLOSSARY

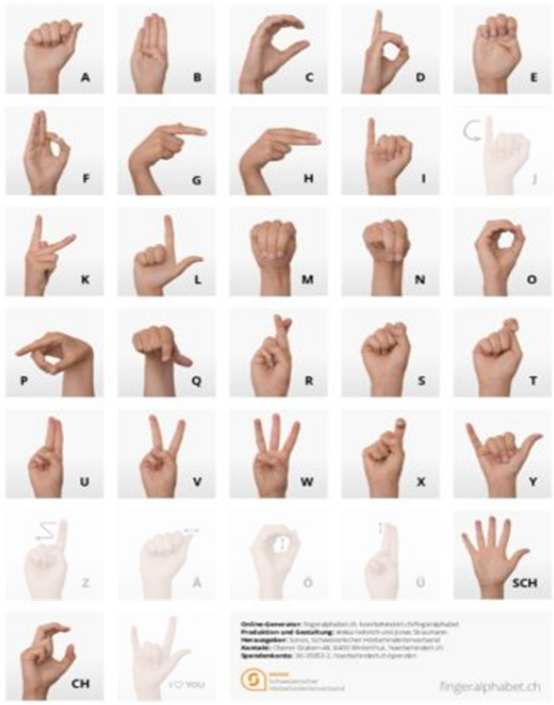
TensorFlow
Is an open source library from google for deep learning. which can be used for training of neuronal networks and is implemented in python and c++;

TensorFlow light
Version of Tensorflow especially designed for mobile devices. It can be used to execute models, but it is not possible to train models. Models can be run on android for example for image classification.

TensorFlow (Light) model
Model is a cluster of Tensors and layers which processed data in the way it was trained for. Light version of this can be run on an android app.

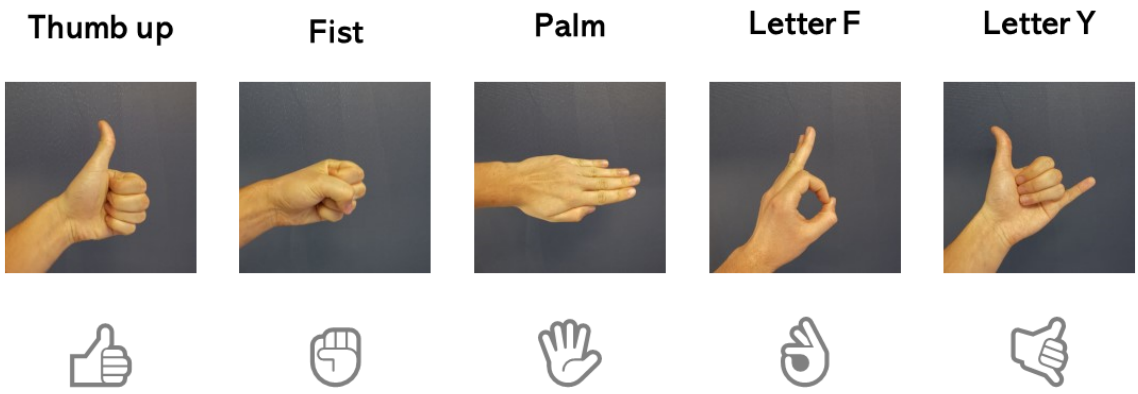
Hand gestures
The 'german' hand alphabet knows 31 'letters'. From there, 5 signals are represented with a motion

component, but this image classifier can only detect static pictures so there would not be possible to detect all letters.



To reduce the complexity I decided to train the model with 5 hand gestures which are easy to recognize.

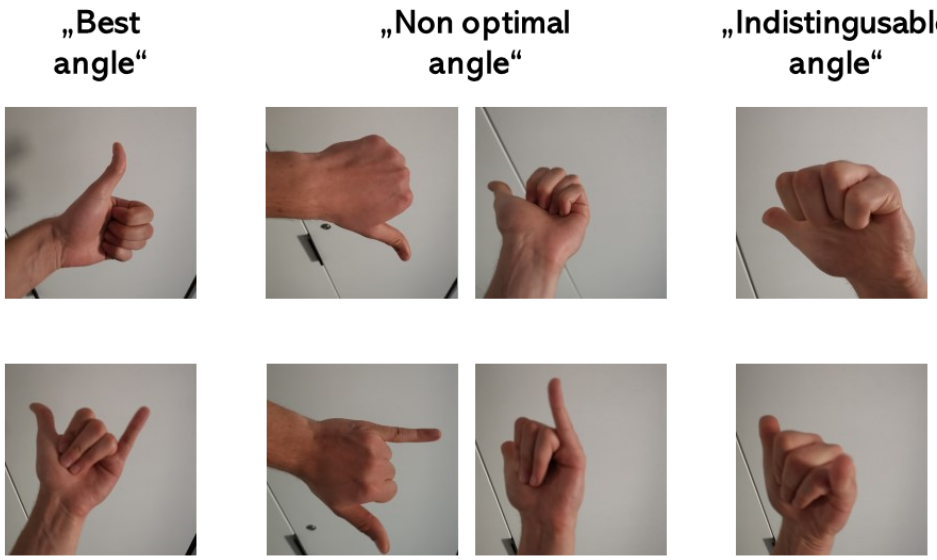
thumb up, fist, palm, letter y, letter f



Perspectives from which the hand gesture can be viewed. I defended 3 categories which I refer to.

- 'best angle'
- 'non optimal angle'

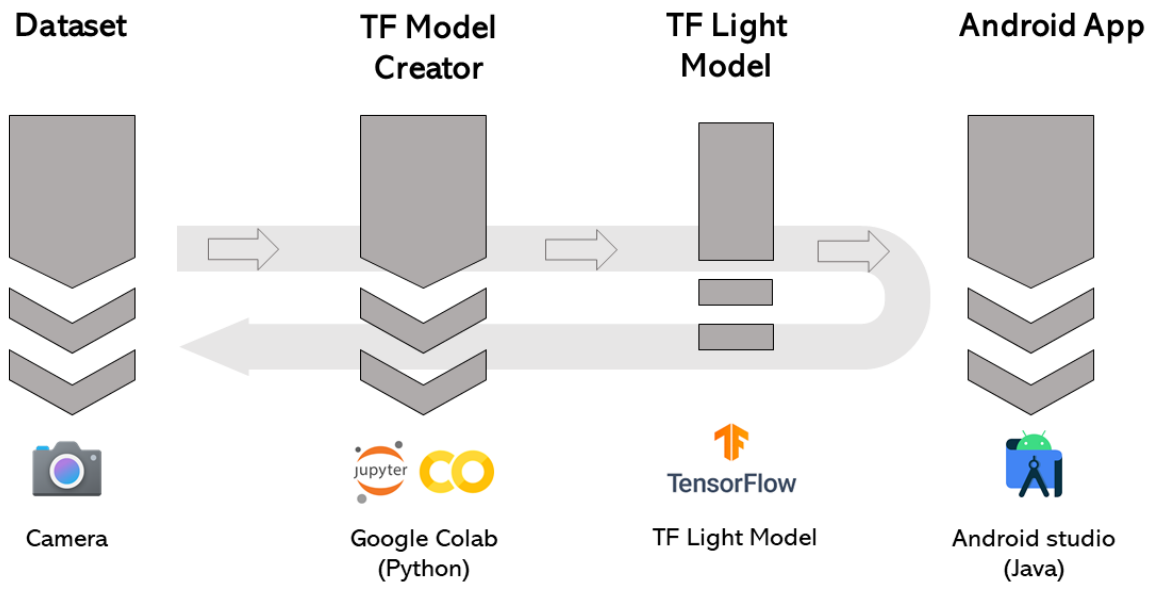
- 'indistinguishable angle'



PROJECT

The project consists of the following main parts.

1. Image library (Data set)
2. Training of the model (Model maker)
3. Android app (App Gesture Detector)



1. Image library (Dataset)

I decided to use my own image library because of the strict quality input limitations for the training process of the model. Second, this provides full control of what gestures are represented and how the gestures are shown. This offers the possibility to take the pictures with the same camera which is used for training and afterwards for recognizing. During the process I gained a lot of experience about which pictures are useful for the training process. I had to make a decision which approach I want to use: if I want to recognize gestures from every perspective or to recognize from the best angle.

Every perspective needs a lot of data. In the experiment I used ~150 pictures for each gesture from multiple angles. This did not lead to a useful model. Recognition on the phone was as low as 10% even for showing the gesture in 'best angle' the reason for this was there are multiple images in the training process where it is impossible to distinguish between the gestures (thumb up, letter y). I assume this problem could be fixed by using a very much larger data set. This works in different models and different sources state that this is not a major problem for AIs. But due to my limitations in efforts and insecure outcome I decided to train the model with images from the 'best angle' perspective.

The second major issue is the background. Humans can be taught easily what the object is by presenting it on a plane background. And the person will recognize this object on any background, because it knows the background does not matter. The image classifier on the other hand does not get any further 'explanation' so it learns and trains itself by finding the similarities between the images. The images which have some consistent parts will be classified as one class, and so represent the same showen object. So it does not distinguish between background and object and therefore it can not ignore the background. This led to a very low success rate after the first experiment on the phone. The data set contained pictures with more or less the same background. In the experiment the background did not match the one from the training images at all and results were very low. So the best way to train the model is indeed by changing the background as much as possible. So the consistent part in the images becomes the object which will be the decisive part and background is insignificant.

Dataset

Examples from dataset, left with mostly uniform background, right with huge variety of backgrounds and settings.

With this knowledge I took pictures of hand gestures with different backgrounds from 'best angle' and 'non optimal angle'. But I removed pictures from 'indistinguishable angle' from the dataset. This proved to lead to the best results. In the end I generated more than 210 pictures for each gesture.

Another huge improvement in recognition was data augmentation before feeding this into the model. Rotation, zoom and brightness adjustment helped a lot.

The pictures were taken in default phone quality and compressed to 180px x 180px for further procedures and storing the data set.

During the whole project the image library was changed and improved all the times after each experiment where I gained new experiences.

2. Training of the model (Model maker)

Training

For training the model I used Google Colaboratory. This so-called web IDE for Python enables users to run python code or jupyter notebooks without setting things up on a local computer. It provides access to Google's Computing power and GPUs for faster execution in building the model. The advantages are, there is no need to install all the project related libraries on the local computer. During the development there is no danger from executing and debugging unknown code from different sources on the internet. Faster execution in building the model. But it also made it necessary to store the dataset in Google drive for easier access. So I reduced the images resolution to 180px x 180px.

I used the TensorFlow Lite Model Maker library. link This Library simplifies training of a TF lite model with custom dataset. It provides transfer learning, data argumentation and different custom settings. With help of this library the amount of training data and the training time can be reduced.

During the project I recently changed the used dataset and the configuration of the model.

Some important variables here are:

- Training and validation data (Dataset)
  Data which is used for training and validation. It is considered a good tradeoff to use a split around 80/20.

- Dropout
  Technique to randomly remove a given amount of neurons in each layer to avoid overfitting.

- Epochs
  One epoch is the full iteration by the learning algorithm over the entire training set. Increasing the number of epochs can achieve better accuracy until a certain level. Too many epochs lead to overfitting.

- Batch size
  Number of samples which is used in a training step.

- Data argumentation
  Input images are modified a bit to represent a larger variety of images. This consists of cropping the image, resizing the image and flipping the image. link

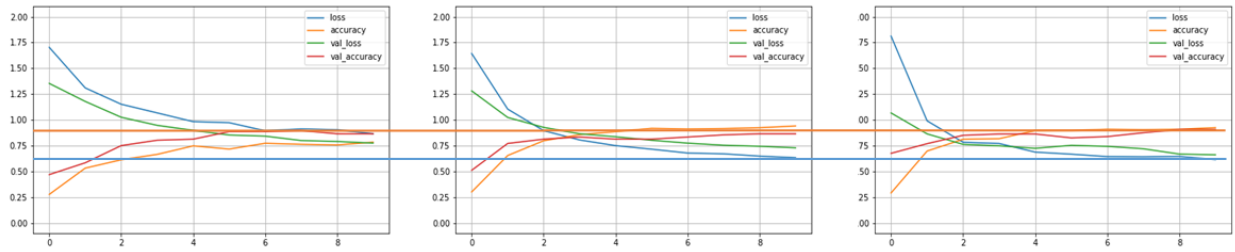
Evaluation

After creating and training the model, the model can be evaluated and if satisfying saved as a TF Lite Model. This TF Light Model can then be integrated in the android app.

To decide the quality of the created model I used different methods.

- The loss and accuracy values of the model.evaluate() method. The lower the loss value and the higher the accuracy value the better the model is. This gives a quantitative overview.

Quantitative evaluation



Plot of ‚loss' and ‚accuracy' over 10 epochs during different training processes. With best values reached in second and third chart.

- For a qualitative impression I plotted a map of 100 random images with the predicted label and the real label. This made it visible in which actual cases the prediction is not working. Are these failures of ambiguous images, where the perspective does not allow any distinction or are there other reasons? This evaluation had a huge impact on how I decided to take the next pictures for the dataset.

Qualitative evaluation

left: false predictions are all considerd as ‚thumb up‘.
right: false predictions are distributed even over all classes.

- Test on the device. After including the TF lite model on the phone the final tests under real circumstances can be done. I tested the recognition capabilities under different backgrounds, lightning and perspectives. The results were changing a lot and sometimes contradictions to the quantitative and qualitative evaluation. But with large enough datasets and the gained knowledge this issue could be solved.

There was a constant feedback loop between taking pictures for the dataset, modifying pictures, customising the model and evaluating the new model. I repeated this with different experiments and settings multiple times. The gained knowledge I used to improve the settings and circumstances but it also led me to define certain limitations.

In the end the best practise for generating a good model was a dataset with a split of 75/25 between training and validation data. A dropout of 20% with 10 epochs and data argumentation.

3. Android app (App Gesture Detector)

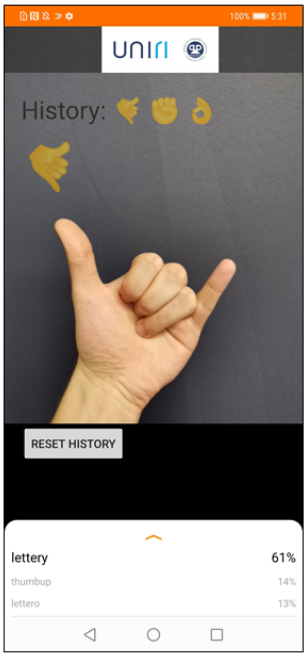
Constantly updated with the newly built model.

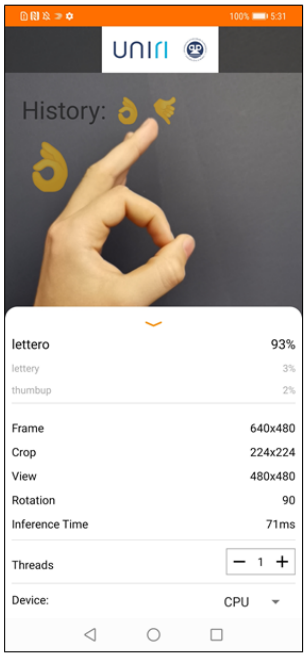
First Idea was to create a java android app from scratch. However during the process I found out that there are a number of existing tutorials and example apps. So an open source example app from TensorFlow was used, which already makes use of the device's camera and sets the basics for the model integration. This app uses a steady stream of images from the camera and has a good reliability while following the android design and coding principles. Doing this on my own would have taken me a lot of time and effort by not even reaching the same quality.

I extended the app with an History feature which shows the last recognised gestures. Furthermore It was necessary to change the models, integration part and several adjustments had to be made.

I did several experiments with the different models under 'real' circumstances. I decided to include the model in the apps data so the model is always available, not depending on network connection.

App Gesture Detector





CONCLUSION

Limitations

limited to the hand alphabet, sign language in whole is more about actions as whole words and needs a body movement detector, this was not part of this project.

But signs which contain a movement could not be detected with this static image classifier.

From some angles it is physically not possible to distinguish between gestures like from 'indistinguishable angle'.

Summary

After increasing the dataset with new images at least 6 times and training a dozen of models under different settings, the project reached an acceptable condition. All five gestures can be detected within milliseconds. However in difficult environments and unusual backgrounds some gestures are not recognized correctly. The dataset used for the best adapted model uses now more than 1200 pictures. Training such a model takes depending on the number of epochs 4-10 min on Google Colabs.

This project is a solid base for further research with larger data sets with hands from multiple persons and more angles. It also lays the foundation to later test his easily on more devices with different cameras.

With more effort it is also possible to create datasets for all signs of the finger alphabet and test this.

Work

From my point of view this is a nice project idea for students. I used a lot of knowledge from the exercises from the Intelligent Systems course. Working in Google Colabs is especially comfortable and straight forward.

But it requires storing data in Google Drive which requires a Google Account. Working with the TensorFlow libraries is nice. It is well documented and there are a lot of examples. This makes it easy to experiment with deep learning algorithms and neural networks.

Files

Source code of app:
AndroidApp_GestureDetector

Example of Dataset of Hands (5 pictures per gesture only):
Dataset_hands

Example TF light model trained with Hands:
Generated_models

Jupyter notebook of Model maker:
modelCreator04

SOURCES

Hand alphabet
https://hoerbehindert.ch/information/kommunikation/fingeralphabet

Base android example app
https://github.com/tensorflow/examples.git

Tutorials:
https://www.tensorflow.org/lite/tutorials/model_maker_image_classification
https://www.tensorflow.org/
https://github.com/tensorflow/examples/blob/83a8b6edfa03fca856b8817c29a06c9d93d4f34b/tensorflow_examples/lite/model_maker

Google Colabs:
https://colab.research.google.com/