
convolutional neural network
Hand Gesture Classification

Project Goals

- ❑ Create own CNN model
- ❑ Train network capable of classifying hand gestures with sufficient accuracy
- ❑ Provide analysis and testing for the model

Tech

- ❑ PyTorch 2.0
- ❑ Python 3.10

Dataset - HaGRID

Original dataset contains 552,992 FullHD RGB images divided into 18 classes of gestures

- For the needs of the program a subset of the dataset is used.
- Subset consist of cropped images of hands only
- Number of images is reduced to ~10k

https://github.com/hukenovs/hagrid

Data preparation

- Consistent size: 64-256px
- Pixel normalization
- Data is splitted onto test(10%) and train data (90%).
- Classes divided in folders on a hard drive
- Data is shuffled, the seed of the shuffle is const.

Two sides of the approach
Two different architectures

TinyVGG

Compact version of VGG, easy to implement, only few layers.

- Straighforward and quick,
- Clear,
- Lightweight,
- Ability to learn NN features,

RasNet18

Powerful deep network, consists of 18 layers. One of RasNet variants.

- Employs residua connections,
- Fully connected layers,
- High accuracy
- Long training time

Usage of two architectures allows for better understanding and greater comparison capabilities

Model 1 - TinyVGG

- Based on VGG architecture with own tweaks
- On this example:
  - 4 convolution layers
  - 2 pooling layers
  - no extra fully connected layers
  - Uses ReLU for activation

```
################################## MODEL ##################################
###########################################################################

class CustomTinyVGG(nn.Module):
    def __init__(self, input_shape: int, hidden_units: int, output_shape: int) -> None:
        super().__init__()
        self.convBlock_1 = nn.Sequential(
            nn.Conv2d(in_channels=input_shape,
                      out_channels=hidden_units,
                      kernel_size=6,
                      stride=1,
                      padding=2),
            nn.ReLU(),
            nn.Conv2d(in_channels=hidden_units,
                      out_channels=hidden_units,
                      kernel_size=5,
                      stride=1,
                      padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2,
                         stride=2)
        )
        self.convBlock_2 = nn.Sequential(
            nn.Conv2d(hidden_units, hidden_units, kernel_size=4, padding=1),
            nn.ReLU(),
            nn.Conv2d(hidden_units, hidden_units, kernel_size=4, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(in_features=30*30*hidden_units, out_features=output_shape)
        )

    def forward(self, mod: torch.Tensor):
        mod = self.convBlock_1(mod)
        mod = self.convBlock_2(mod)
        mod = self.classifier(mod)
        return mod
```

Model 2 – RasNet18

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

tures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of block

Source: https://pytorch.org/hub/pytorch_vision_resnet/

- RasNet variant with 18 layers in total
- Fully connected layers
- Also uses ReLU for activation
- Normalization to stabilize training process using BatchNorm2d

Trainig Process

Parameters that were adjusted to get best effect:

- Learning rate
- Optimizer algorithm
- Batch size & Epochs
- Number of Convolutional layers
- Padding and Kernels sizes
- Dropouts
- Number of Pooling layers
- Activation functions
- Weight Decay
- Image dimensions
- Overall model complexity

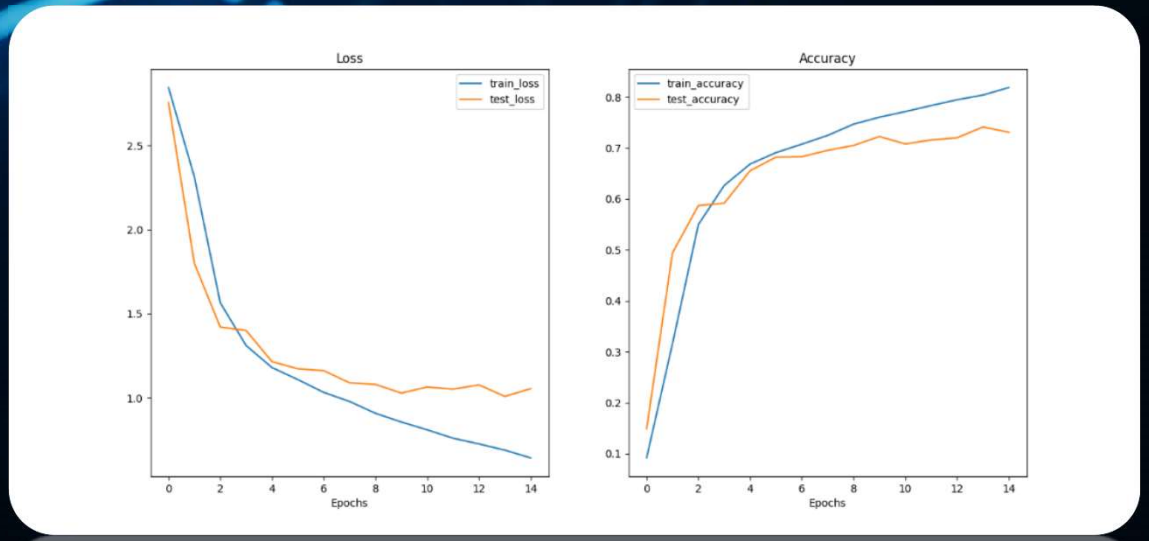
Consts

- ❑ Seed
- ❑ Dataset Split
- ❑ Device (GPU)
- ❑ Image channels
- ❑ Classes
- ❑ Loss function

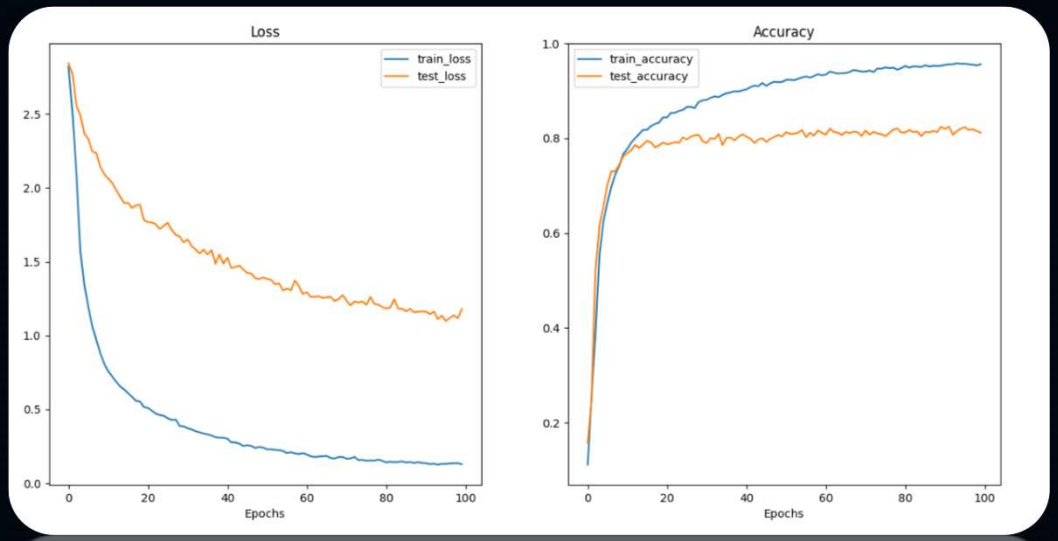
Trained on up to ~10000 images

Current Results

TinyVGG



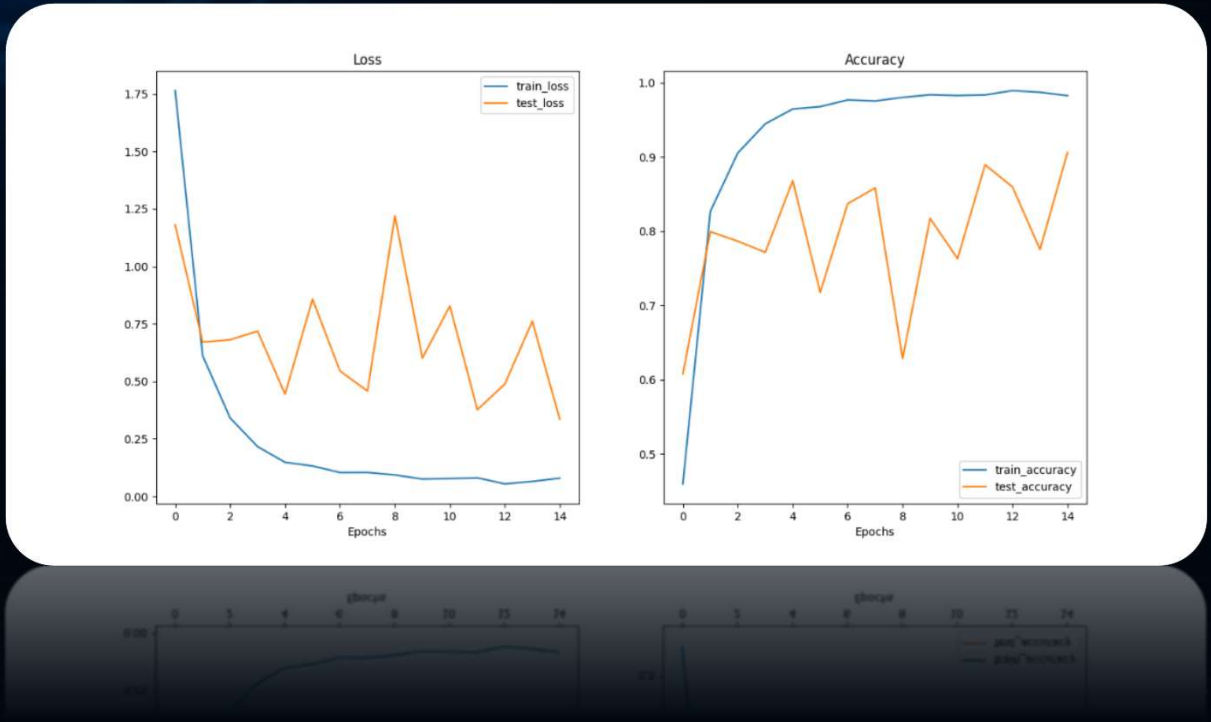
~75% accuracy



~80% accuracy

Current Results

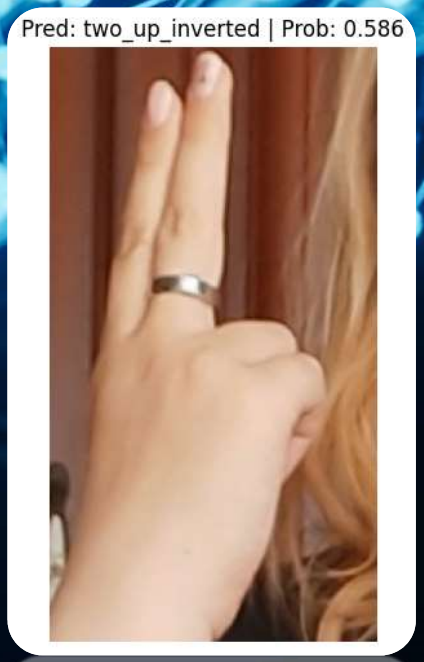
RasNet18

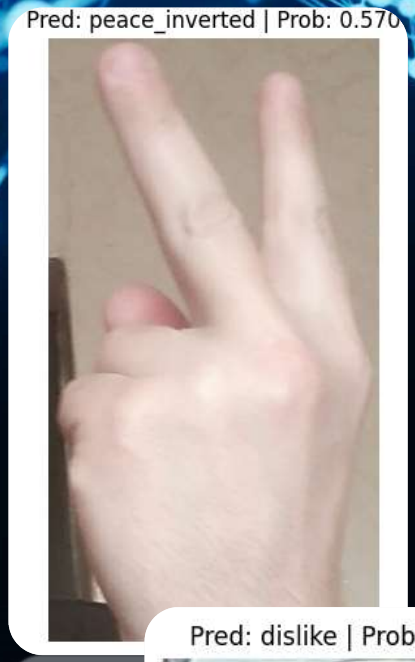


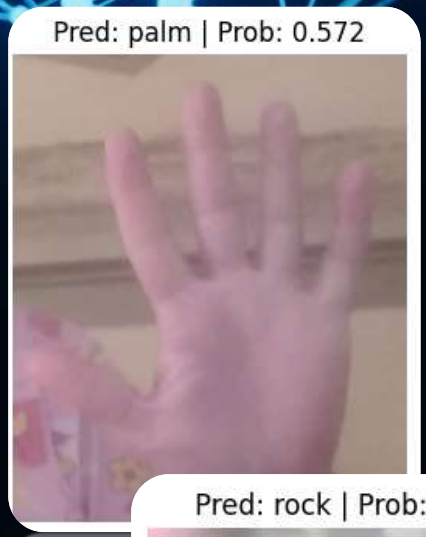
~90% accuracy

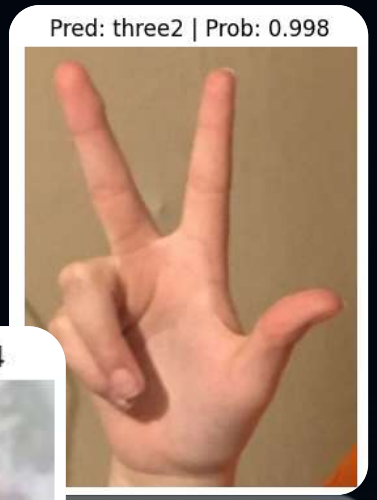
Examples

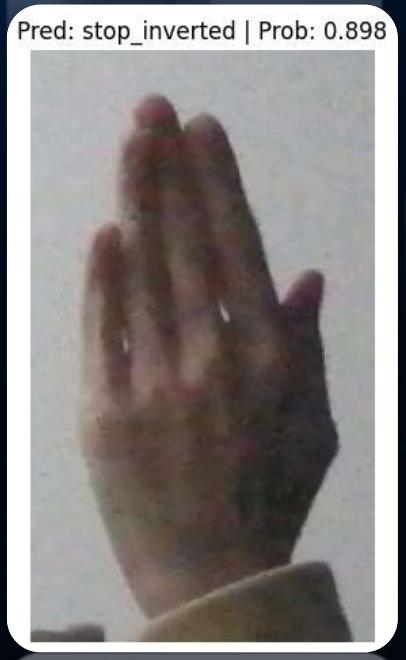

Pred: fist | Prob: 0.986


Pred: two_up_inverted | Prob: 0.586


Pred: peace_inverted | Prob: 0.570


Pred: palm | Prob: 0.572


Pred: three2 | Prob: 0.998

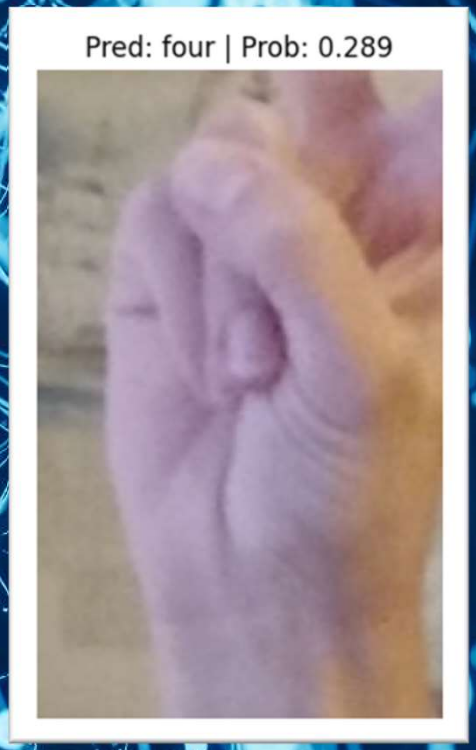

Pred: stop_inverted | Prob: 0.898

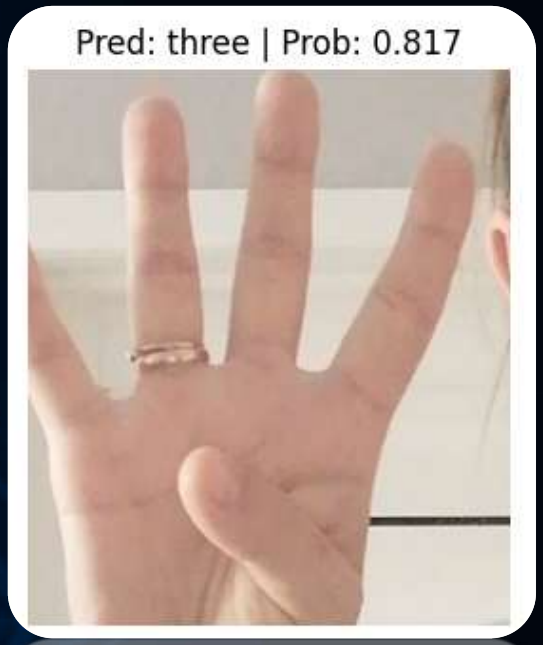

Pred: call | Prob: 0.997


Pred: dislike | Prob: 1.000


Pred: rock | Prob: 0.644

Not so accurate guesses





<< Faulty data

The End

Some hands-related proverbs

Two watermelons can’t be held in one hand. ~ Afghan Proverb

A gentle hand may lead even an elephant by a single hair. ~ Iranian Proverb

You can’t use your hand to force the sun to set. ~ Nigerian Proverb

Promises have legs. Only a gift has hands. ~ German Proverb

by: Adrian Zaręba