

Comparing Handwritten Number Recognition Networks Keras versus Nengo

Simon Tröster, Dominik Weidner

23.01.2024

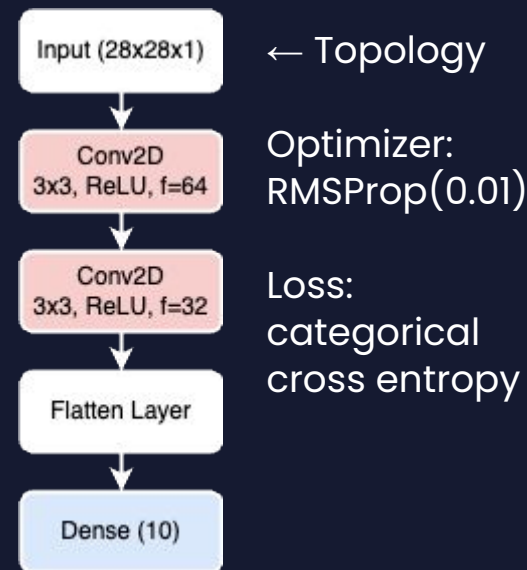


Project Overview

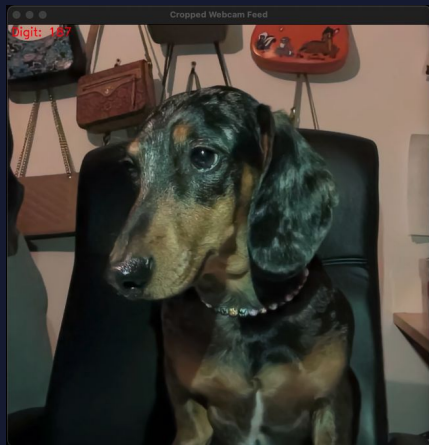
- Create 2 versions of a handwritten number recognizer
- Utilizing the MNIST dataset for training
- Provide model input via webcam
- Compare the 2 approaches regarding performance, programmability, usability and other metrics

Traditional Approach: CNN with Keras

- Straight forward implementation
 - Well documented resources and examples
- This approach was used to test and implement the webcam functionality
 - More sophisticated approach by finding contours in the full webcam image didn't prove viable
 - → Solution: Using the biggest square image frame possible



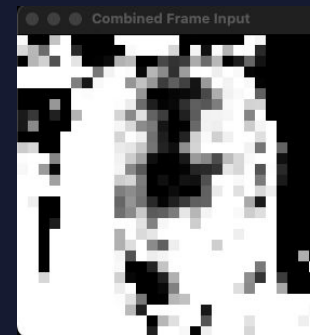
Model Input Preprocessing with Webcam



1080p raw capture



Transformations:
greyscale
gaussian blur



Inverted model input
28x28p (280x280)

Model Input Image Buffering

- Running the model/sim every frame is expensive
- Instead: A 30 frame buffer is implemented

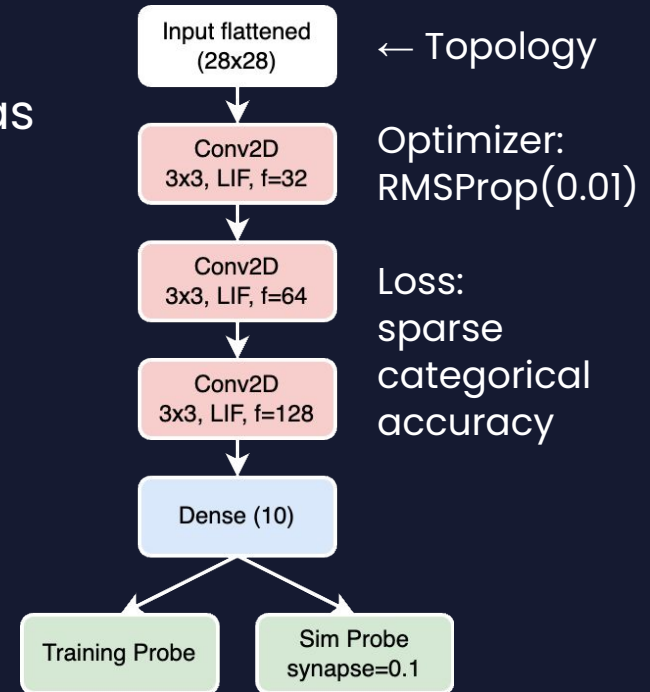


This reduces camera shake and increases general image quality

In Nengo, this is needed as the simulation can not yield “real-time” results

The Nengo Approach

- Goal: Mirroring the Keras CNN as closely as possible but utilizing spiking neurons (LIF)
- Additional goal: Incorporating the “real-time” webcam functionality
- Additional Conv2D layer added as it improves result quality



The Nengo Approach – Requirement Conflicts

- nengo, nengo-dl and tensorflow library are needed
- Conflicts: Version and system compatibility are problematic on mac & windows w/o nvidia gpu



Working on Windows (with NVIDIA):

```
tensorflow-gpu==2.11.0  
nengo==3.2.0  
nengo-dl==3.6.0  
(CUDA-toolkit 11.2 & cudnn 8.1.0)
```



Working on Mac M2 Pro:

```
(Python 3.9)  
tensorflow-macos==2.8.0  
tensorflow-metal==0.4.0  
nengo==3.2.0  
nengo-dl==3.6.0
```

Results – Both Models in Numbers

Keras CNN



Training time: **~4 min**

(batch_size=32, epochs=10)

(batch_size=300 → <3min but **OF**)

Accuracy after training: **98.71%**

Prediction time: **<1s** (~36ms step)

Nengo SNN



Training time: **~10 min**

(batch_size=300, epochs=10)

Training goal: Similar acc. as the CNN

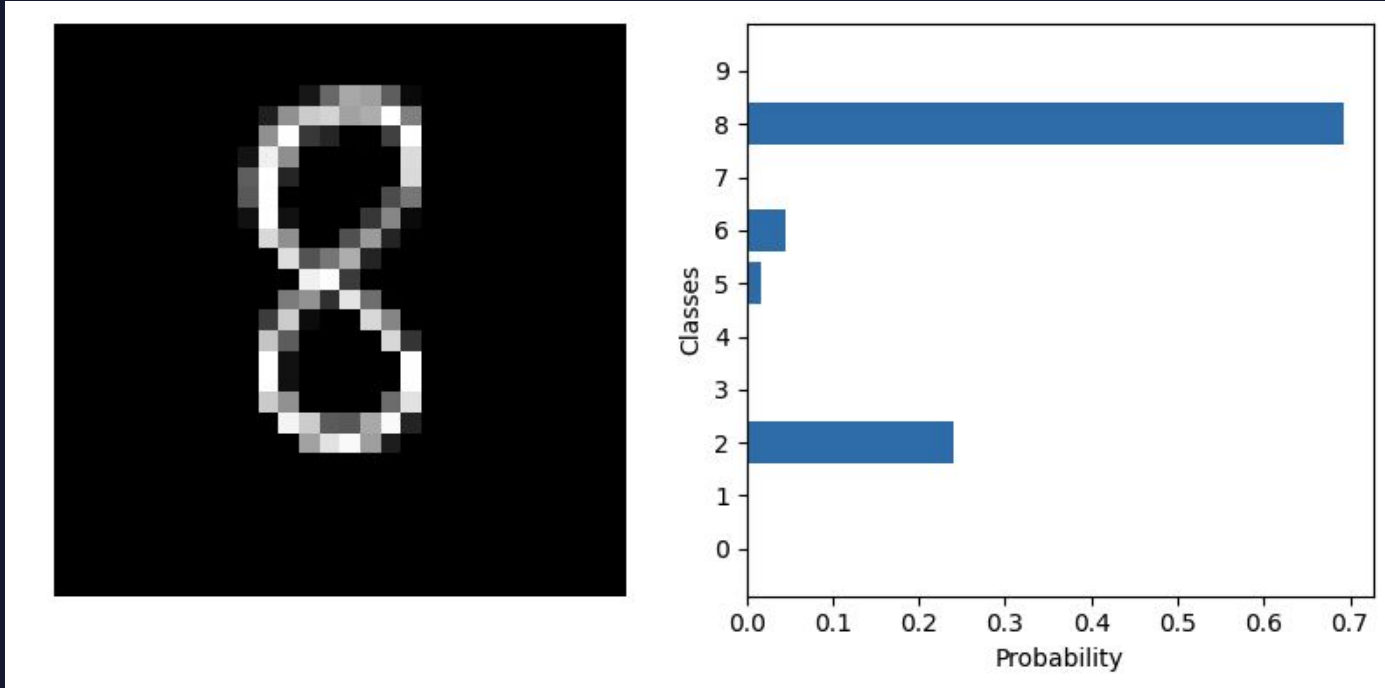
Accuracy after training: **98.72%**

Simulation time: **~10s** (30 steps @ 0.001)
→ Maybe multithreading would help

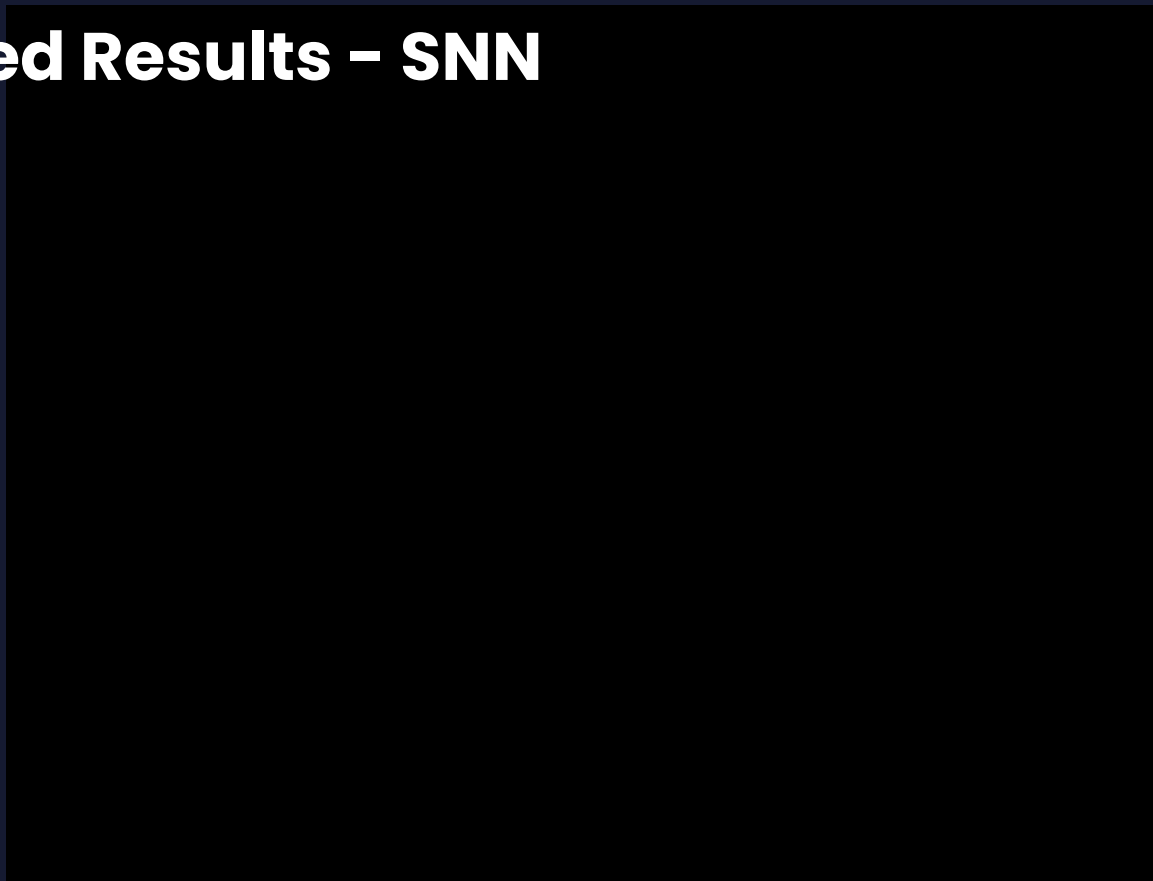
Visualized Results - CNN



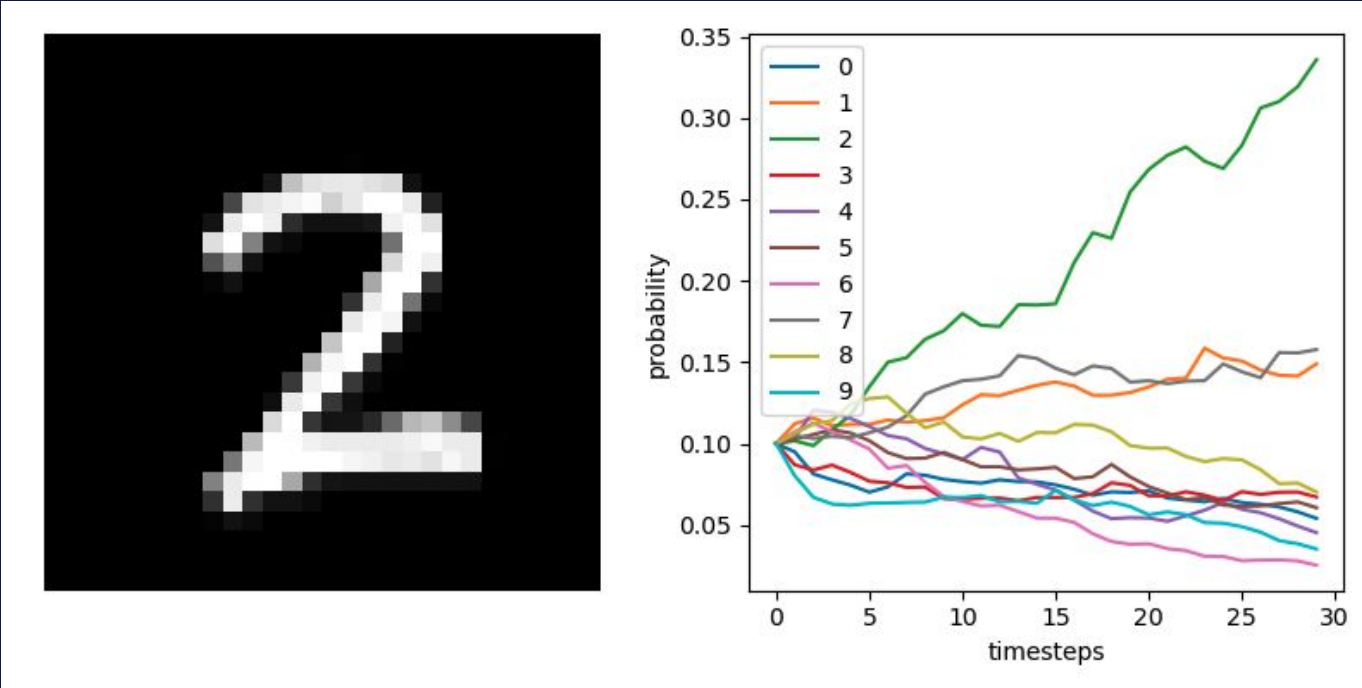
Visualized Results – CNN Plotted



Visualized Results – SNN



Visualized Results – SNN Plotted



Conclusion

- It is possible to create a well-functioning handwritten digit recognizing SNN with webcam functionality
- Obvious drawbacks include longer training times and the simulation time
- In the right circumstances (low power environments) the SNN can provide a viable alternative to traditional models

Outlook

- It would be interesting to see the SNNs performance compared with an implementation on neuromorphic hardware (Intel Loihi)

Thank you for your attention!

