Project summary: Programming project: Interactive Dreaming: Task C
By Rozalina Petkova, Lotta Piefke and David Masurek

The common opinion about sleeping and dreaming exclude conscious states of mind and with this the possibility of two-way communication with a sleeping person. However, research on the topic reveals something different. Back in the 1970s and 1980s, two independent studies showed that lucid dreamers could communicate with researchers by means of previously agreed upon eye movements when they were in REM sleep. More recent studies from Netherlands, Germany, France and USA also succeed in the real-time communication with lucid dreamers, using volitional control of gaze direction or of different facial muscles (Konkoly et al., 2021 & Stickgold et al., 2021). More research in the area would mean more complex ways of communication. One such option is the usage of eye-swiped 2D messages.

The general goal of the project was to decode a given eye-swiped 2D message, potentially from a lucid dreamer, into words. To accomplish this, we first programmed a CNN model to recognize letters using tensorflow. Once we successfully implemented this, we wanted to build a subsequent RNN that can generalize to a sequence of letters and would therefore be capable of detecting words.

The notebook for the CNN model we built which can recognize letters can be found here. We used the emnist dataset (Cohen, Afshar, Tapson and Schaik, 2017). After 30 Epochs of training we reached a testing accuracy of ~ 0.9204. With this architecture working, we moved on to words.

For the dataset we chose the IAM Dataset, a dataset with handwritten images and labels (Marti & Bunke, 2002). However, we soon noticed that the whole data preprocessing and model evaluation was not manageable for us in the given project time, since we are all very new to tensorflow and modeling neural networks. We were happy to find an article (linked here) which detailed the necessary preprocessing and evaluation methods for the dataset and given task. This gave us more time to develop a good model with the best possible results.

We compared the following models:

1. The original model presented in the article. Regarding the parameters we later changed and built upon, this model included two convolutional layers with the kernel initializer “he_normal” and one dropout layer.
2. A model with four convolutional layers (inspired by our letter recognition model).
3. A model with four convolutional layers, batch normalization and an additional dropout layer to deal with potential overfitting as a result of our model of higher complexity.
4. A model with four convolutional layers, batch normalization, an additional dropout layer and the kernel initializer “RandomNormal”, which is the initializer we had previous working experience with.
5. A model with four convolutional layers, two dropout layers, and the random initializer

To evaluate the models we let each of them run for 50 epochs and then compared the testing loss as well as the edit distance, which is a good metric to evaluate models such as ours which are using CTC loss. The edit distance describes the number of character changes necessary to transform the predicted label to the target label. If the edit distance is 0, prediction and target label are identical.

We are happy that, as can be seen in Fig 1., all of our adjusted models performed better than the original model. The 2nd model has the lowest training loss (Fig 2), but not the lowest testing loss (Fig 3), as it overfitted (Fig 4), which was expected. In the 3rd model, where we applied counter-measures, there was less overfitting and the test loss, as well as the edit distance, was the lowest. There was no overfitting in both the original model (1) and the 4th model. Since the only difference between the 3rd and 4th model is the kernel initializer, it seems that in this specific case the “he_normal” initializer works better, but is more prone to overfitting. The 5th model performs worse than the 4th with the only difference being the batch normalization, which shows that batch normalization makes the performance better. It would be interesting to see the results of the 4 networks after more than 50 epochs, but unfortunately we did not have the time for this. Fig 5 shows a sample prediction of Model 3.

Figures:

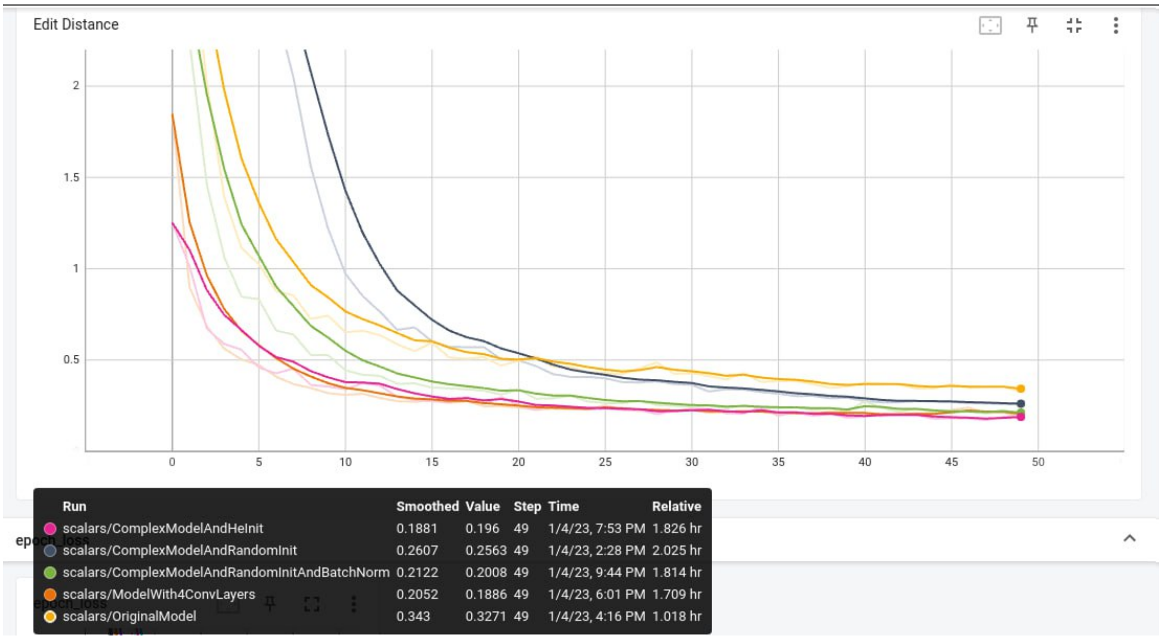

Fig 1: Edit Distance of the 5 models

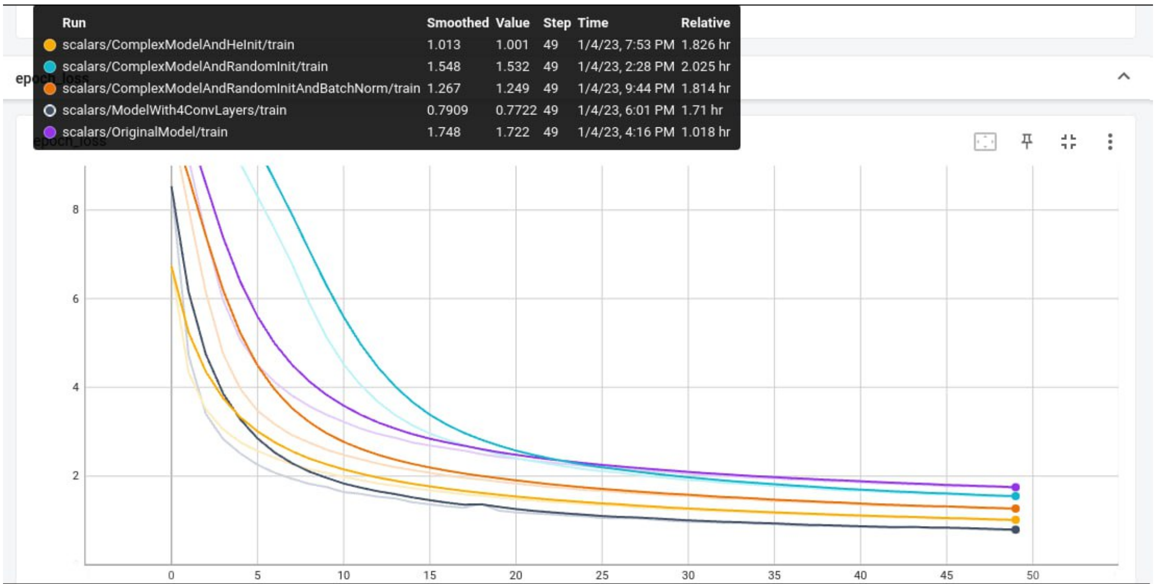

Fig 2: Training Loss of the 5 models

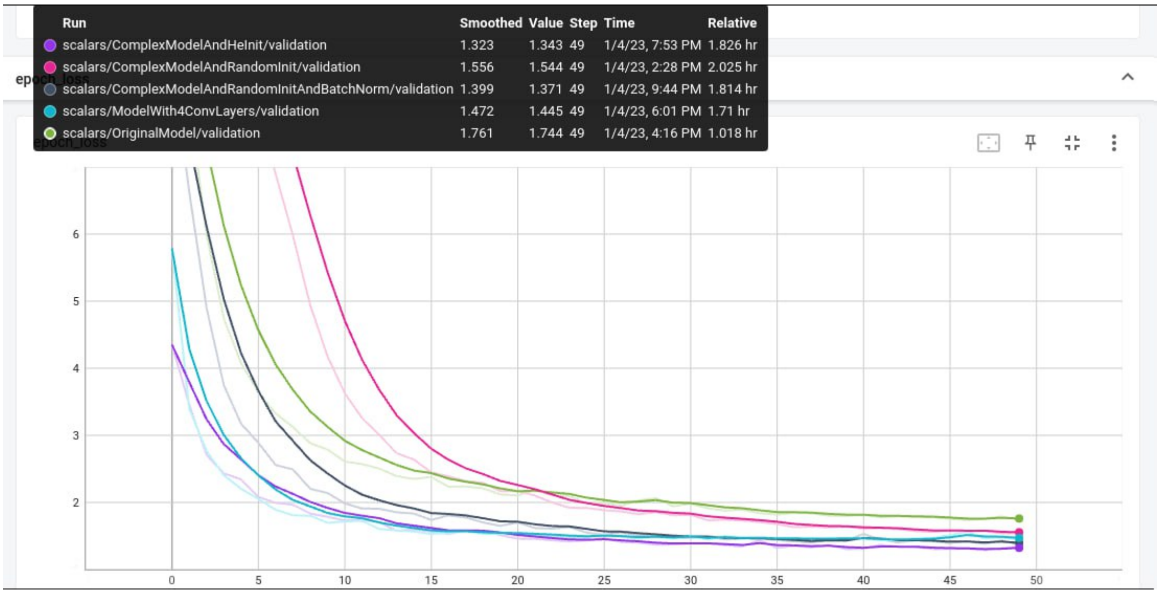

Fig 3: Test Loss of the 5 models

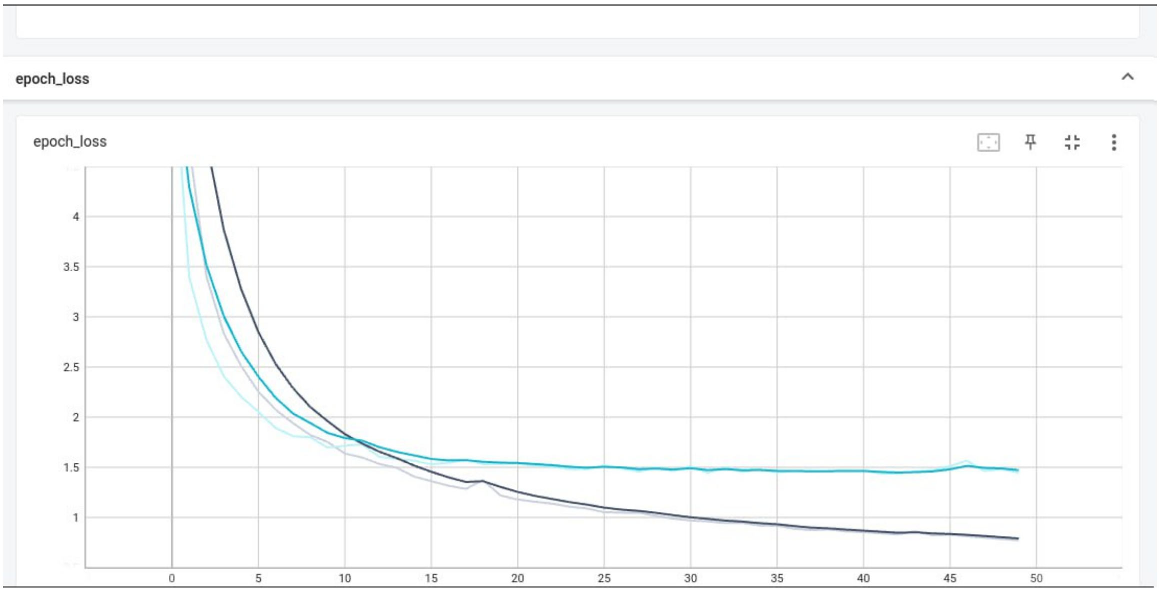

Fig 4: Overfitting of Model 2

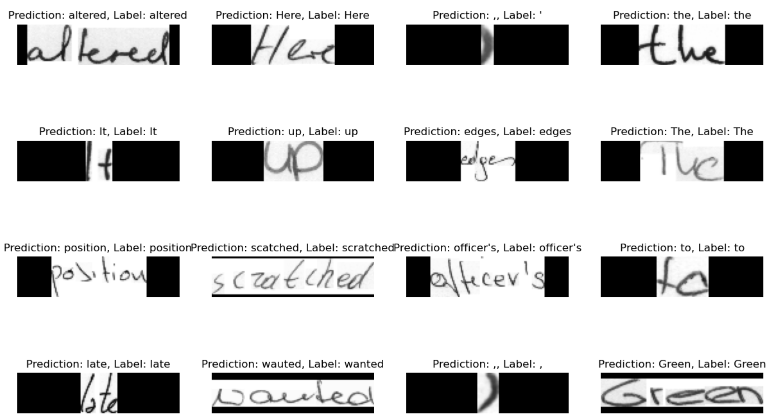

Fig 5: Predictions and Labels on Validation Set by Model 3

Sources:

G. Cohen, S. Afshar, J. Tapson and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2921-2926, doi: 10.1109/IJCNN.2017.7966217.

Marti, UV., Bunke, H. The IAM-database: an English sentence database for offline handwriting recognition. *IJDAR* **5**, 39–46 (2002). https://doi.org/10.1007/s100320200071

Konkoly, K. R., Appel, K., Chabani, E., Mangiaruga, A., Gott, J., Mallett, R., Caughran, B., Witkowski, S., Whitmore, N. W., Mazurek, C. Y., Berent, J. B., Weber, F. D., Türker, B., Leu-Semenescu, S., Maranci, J.-B., Pipa, G., Arnulf, I., Oudiette, D., Dresler, M., & Paller, K. A. (2021). Real-time dialogue between experimenters and dreamers during rem sleep. Current Biology, 31(7). https://doi.org/10.1016/j.cub.2021.01.026

Stickgold, R., & Zadra, A. (2021). Sleep: Opening a portal to the dreaming brain. Current Biology, 31(7). https://doi.org/10.1016/j.cub.2021.02.016

https://keras.io/examples/vision/handwriting_recognition/

Our Code and the video explanation:

The CNN model for letters:
https://github.com/lolotta/SleepTechWordDetection/blob/master/letters.ipynb
The Model for words:
https://github.com/lolotta/SleepTechWordDetection/blob/master/PredictsOurSamples.ipynb
The whole github repository: https://github.com/lolotta/SleepTechWordDetection

A Video explanation of our project can be found here: https://youtu.be/Y8jr16BuUyI or https://myshare.uni-osnabrueck.de/f/04e2365f522f4bcd992c/