Project Report: Convolutional Neural Network and Recurrent Neural Network for Earthquake Detection and Localization

Till Beemelmanns

6. December 2018

1 Introduction

In order to improve seismic hazard assessment recent publications proposed machine learning and deep learning methods to detect and locate earthquakes. These approached are based on publicly available dataset that contain continuous waveform measurements and corresponding earthquake events. The most recent and most popular paper that applied deep learning to this kind of data was published by Perol et al. [4]. The researchers use Convolutional Neural Networks (CNNs) on waveform data to detect earthquakes and location. In the course of this project the results in [4] using a CNN architecture have to be validated and in addition the prediction accuracy should be further improved. The application of Recurrent Neural Networks (RNN) using Long-Short Term Memory Cells (LSTM) [2] were applied to this time-series classification problem, but proved to be ineffective. Finally, a fused architecture consisting of CNN layers and bi-directional LSTM cells was applied. A benchmark of both architectures on the same training and test datasets is performed.

2 Dataset and Approach

The 70 GB dataset [4] contains continuous waveforms and earthquake events and was captured from February 2014 to November 2016 in Oklahoma (USA). The overall dataset contains 2709 earthquake events that were captured by two seismic stations. In Appendix B some of the data samples are visualized. The *test* dataset is defined as the waveforms and events that were captured during *July 2014*. It contains 209 events. The *training* dataset consists of all other measurements.

The general approach of the earthquake localization is to discretize the domain in clusters and perform the localization on this simplified domain. As shown in Appendix A, the events are clustered on basis of their latitude and longitude. The k-Means clustering algorithms was used to determine 6 clusters. In order to train the neural network, the continuous waveform was divided in windows of 10s length with sample rate of 10Hz on 3 channels. Hereby, event windows contain an earthquake event and noise windows do not contain any event, just noise. Further, for every event window the correct cluster (label) is known. That leads to a classical classification problem. Given a random 10s window of seismic measurements, we want to find out in which cluster that event happened.

3 Evaluation of ConNetQuake

One of the goals of this project is to verify the results of Perol et al. [4]. Hence, the training of the ConvNetQuake network [3] has been conducted on a new preprocessed training dataset. The original ConvNetQuake consists of 8 plain convolutional layers with kernel size of `3x3` and a stride of `2x2`. Pooling layers have not been applied to this network. Figures 8a and 8b indicate that the neural network converges approximately after 20.000 training steps. In this last state of the training, the model was evaluated on the test dataset (see Listing G) obtaining a location accuracy of 75.3 % using 6 location clusters. This corresponds approximately to the accuracy of 74.6 % reported in [4]. The slight difference might occur due to the selection of the test dataset and at which training epoch the model was stopped. Further, the detection of an earthquake could be detected with almost 100% accuracy, same accuracy as reported in [4].

4 Evaluation of LSTM-CNN

During the development of this project new architectures were added to the existing ConvNetQuake framework. A raw LSTM architecture and a combination of Long-Short-Term Memory Cells and Convolutional Neural Networks were deployed in the hope that the LSTM could further improve the prediction quality. In several research papers, LSTMs archived state of the art performance in sequence processing tasks. Figure 10 shows that the existing `ConvNetQuake` class was inherited in order to implement the new architectures.

In the first attempt, a raw stacked bidirectional LSTM model was deployed on this setting. But it turned out that the convergence of this models was very slow and even after long training the prediction quality was bad. A possible explanation for this behavior can be found in the lengths of the training windows. Each window has a lengths of 10s which leads to a training tensor sample of 1000×3 using a sampling frequency of 10 Hz on 3 channels. With this high number of time steps a raw LSTM encounters the problem of vanishing gradients. The number of time steps in this setting is simply to high in order to backpropagate the error gradient from the output layer to the weights. A down sampling of the input sequence was applied which lead to better but still not satisfactory results.

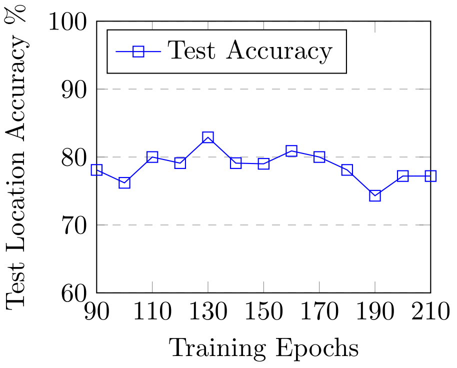


Figure 1: Test Location Accuracy over Training Epochs using the CNN-LSTM model.

In the second attempt, the LSTM architecture was combined with convolutional layers. The LSTM-CNN consists of two feature extraction layers as an the input of the neural network. They have the same shape as in the original model. The output of the convolutional layer was then feed into an array of 5 Bidirectional LSTM Cells. In the final layer a fully connected classification layer was deployed. This architecture is visualized by Figure 7.

After a simple hyper parameter search a good set of train parameters was obtained. As shown in Figure 1 and Listing G the model could obtain a maximal location accuracy of 82.9 % on the *test dataset.* This instance was achieved using 130 training epochs and results of that model are show in Appendix E. The neural network computes the probability over the different clusters given an unseen test input window and in most cases the prediction was correct. Choosing the training epochs in range 110-170 leads in general to good results with an approximate average of 80% location accuracy. Further training leads to overfitting of the model.

5 Conclusion & Answering of the Research Questions

- **Can the results of Perol et al.** [4] **be verified ?**
  Yes, the same prediction quality as in the paper could be obtained.

- **Has a RNN-LSTM a better performance than a CNN using continuous waveform data ?** A pure RNN-LSTM architecture could not perform better than a CNN architecture. The sequences length of 1000 samples might be too long for recurrent neural networks. They encounter the problem of vanishing gradients over time. Downsampling the input sequence helped only a little. However, a fused architecture consisting of convolutional layers and stacked LSTM cells could improve the localization prediction from 74.6 % to 82.9%.

- Can a **different dataset** (e.g. Hi-Net [1]) be used for the same methodology ?
  Yes. Requirements of the dataset are as following: We need a catalog file that contains event time, and the estimated locations for each of the earthquakes. Further, we need continuous waveform measurements of at least one seismic station that is close to those events. It is advisable to convert all the catalog files and the continuous waveform dataset into the same format as in [3], because all preprocessing steps rely on that format.

- How can we perform a prediction on a **continuous spatial domain** ?
  Instead of clustering the spatial domain and performing a classification over the different clusters, we could try to predict the coordinates directly. For this regression task, we should use a *Mean Squared Error* as a loss function.

6 Outlook

- So far the classification was performed on a 2-dimensional plane. However, the depth of earthquakes also plays an important role. One could perform the k-Means clustering of the earthquakes in 3 dimensions, adding the depth along with longitude and latitude coordinates.

Appendix A Earthquakes and Seismic Stations

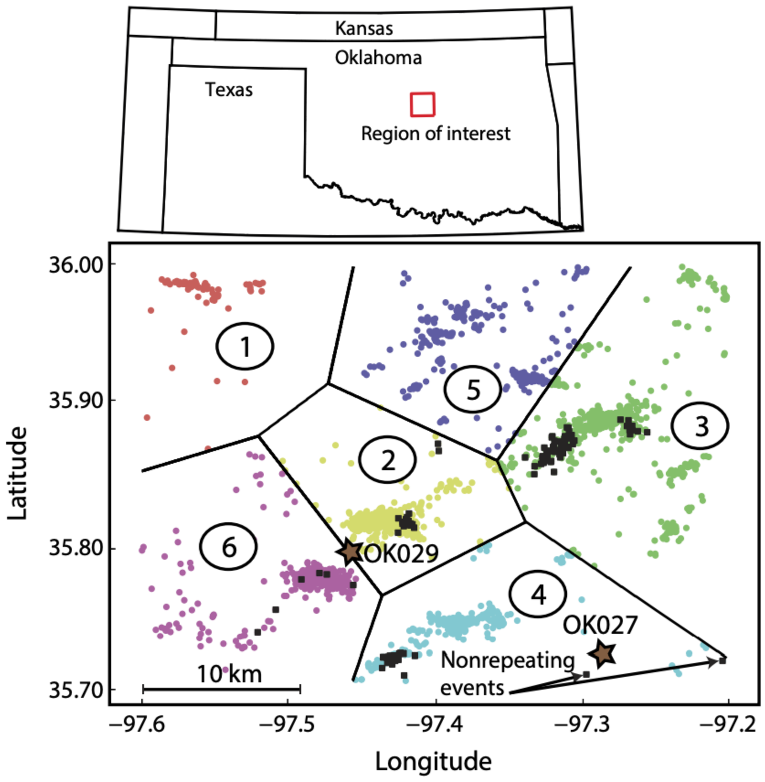


Figure 2: Two dimensional representation of the earthquakes and the clustering of the k-Means algorithm. Each earthquake was assigned to one of the 6 clusters and the neural network predicts the probability over the different clusters for each input sample. The two stars represent the two seismic stations that captured the continuous waveform data. The clustering in this thesis is slightly modified since the clustering algorithm was run again. Figure is taken without any modifications from [4].

Appendix B Training Dataset Samples

The following training dataset samples have been captured in July 2015. Each snippet is 10 seconds long and the event is shifted by two seconds. These are positive examples, but the training-set also contains negative examples without any event.

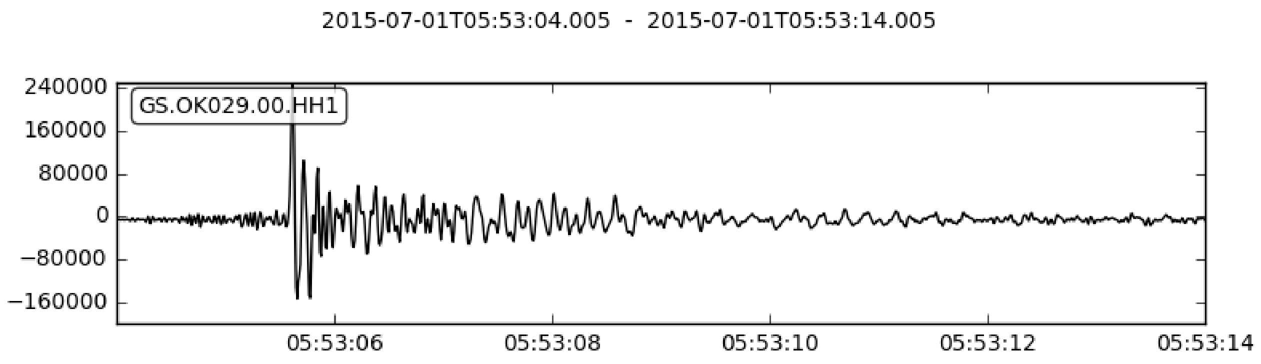


Figure 3: Positive Event - Waveform of Event 0 in Cluster 3

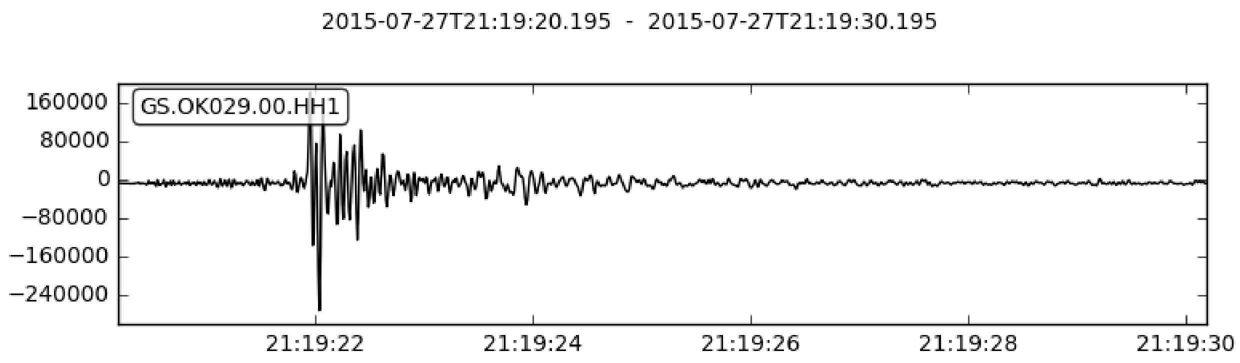


Figure 4: Waveform of Event 78 in Cluster 3

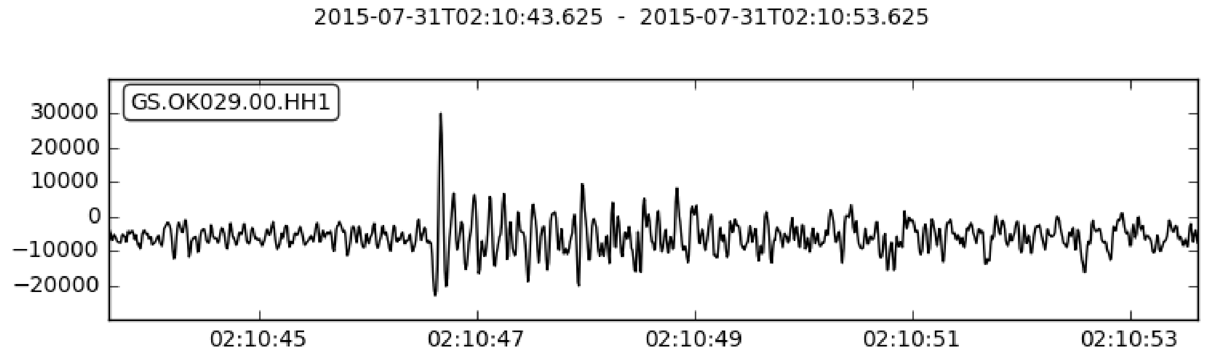


Figure 5: Waveform of Event 95 in Cluster 0

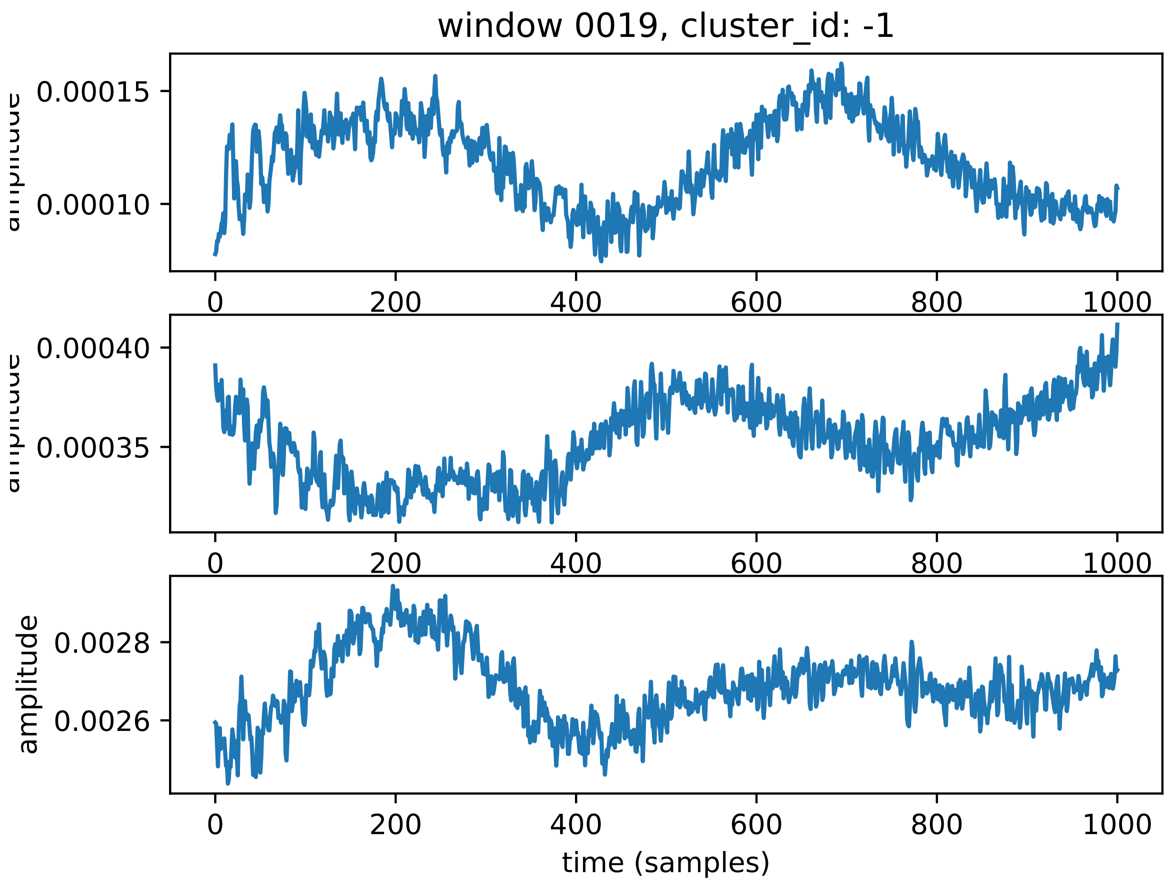


Figure 6: Waveform of a negative time period of 1000s which corresponds to Cluster -1 (no cluster)

Appendix C Network Architecture

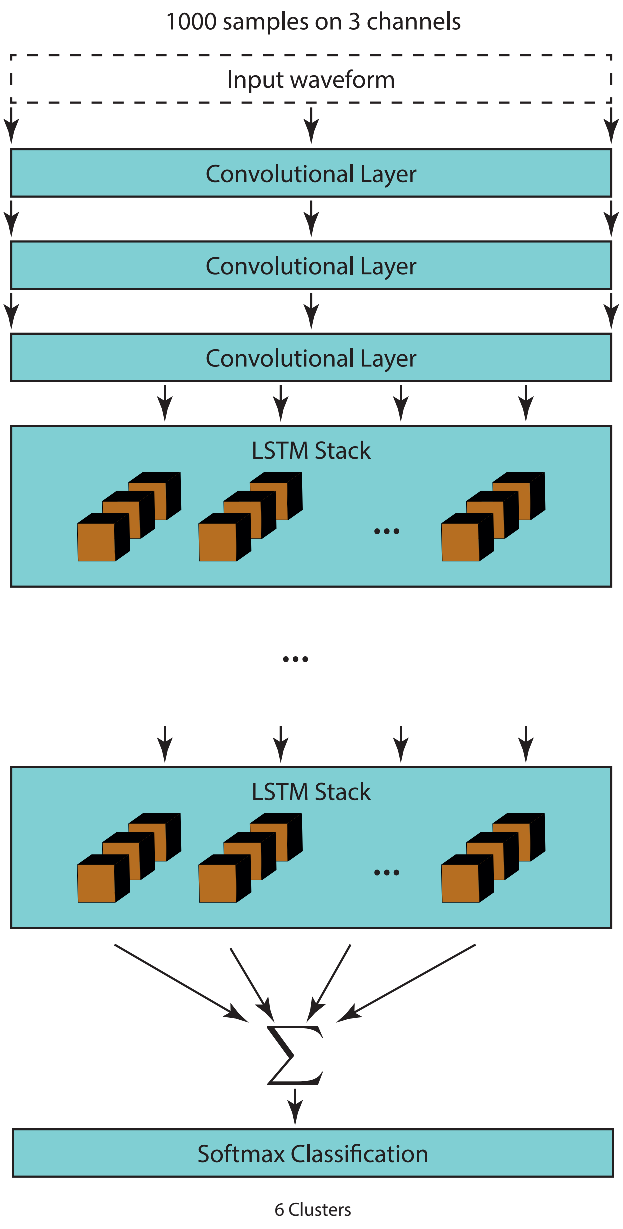
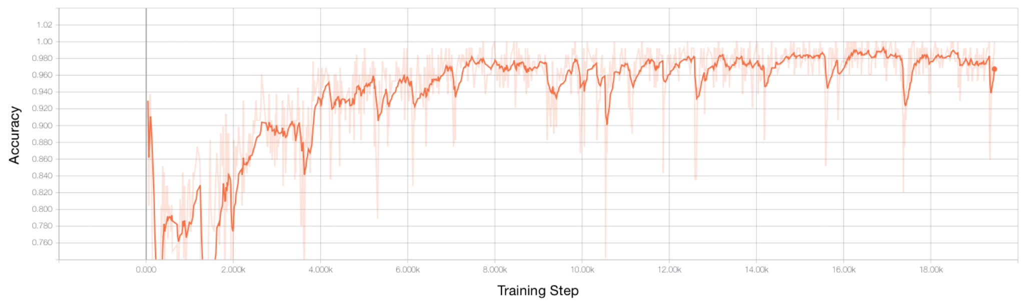
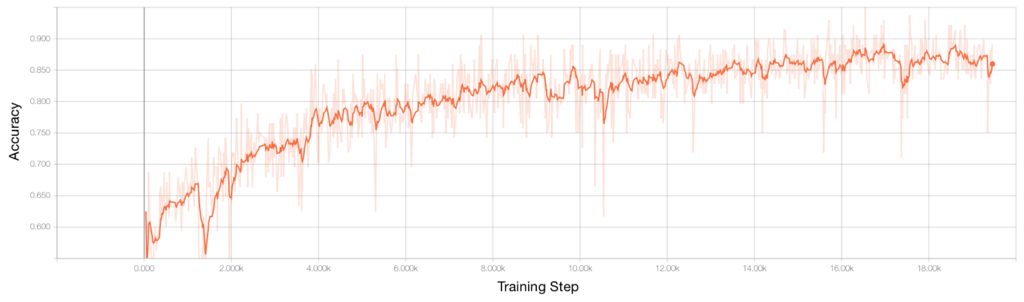


Figure 7: The network consists of Convolutional Layers and LSTM Layers. All layers are fused and trained together. 5 LSTM stack layers were deployed. The architecture and figure are inspired by a Tensorflow tutorial [5].

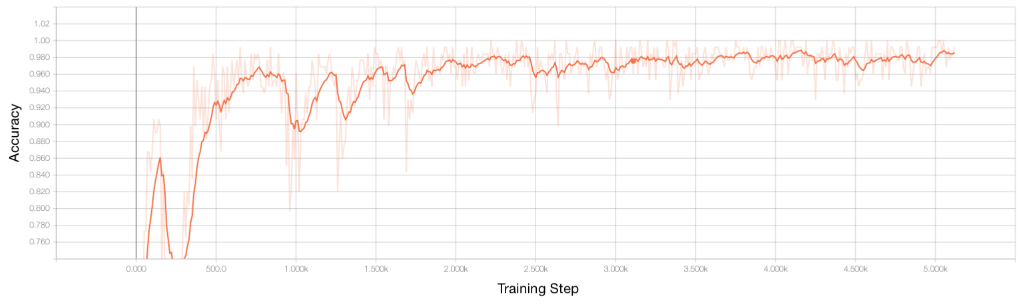
Appendix D Training of the Neural Networks



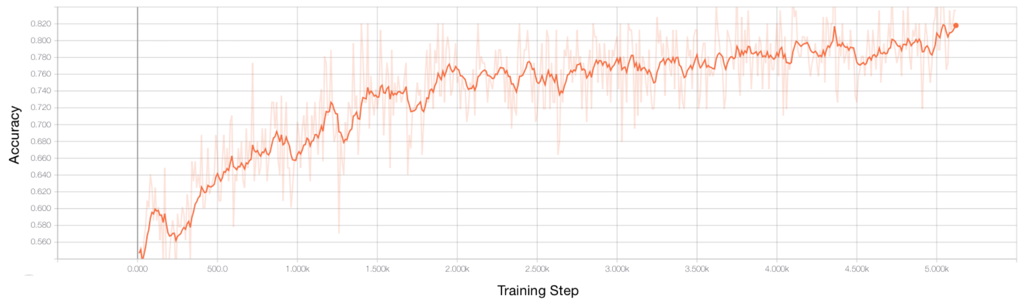
(a) **ConvNetQuake**: Evolution of the Earthquake Detection Accuracy over Training Steps



(b) **ConvNetQuake**: Evolution of the Localization Accuracy over Training Steps

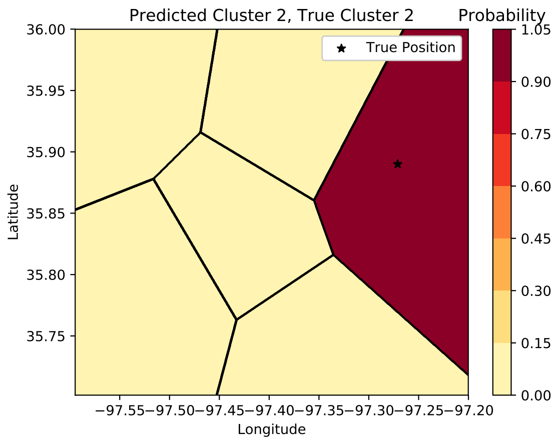


(c) **CNN-LSTM**: Evolution of the earthquake detection accuracy over training steps

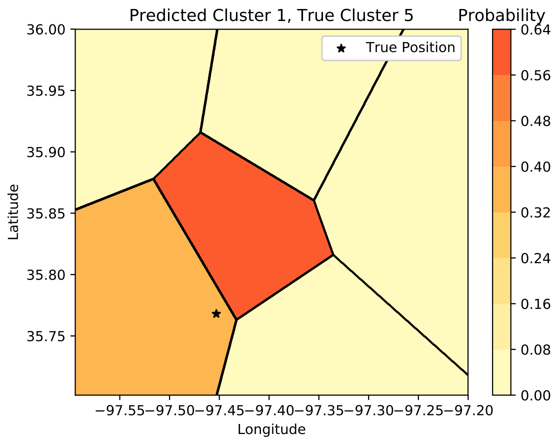


(d) **CNN-LSTM**: Evolution of the earthquake localization accuracy over training steps

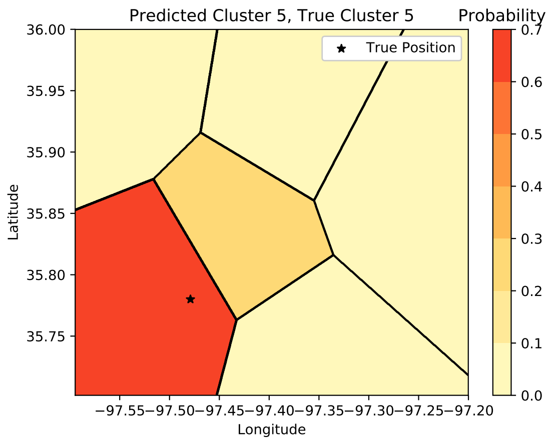
Appendix E Prediction Samples



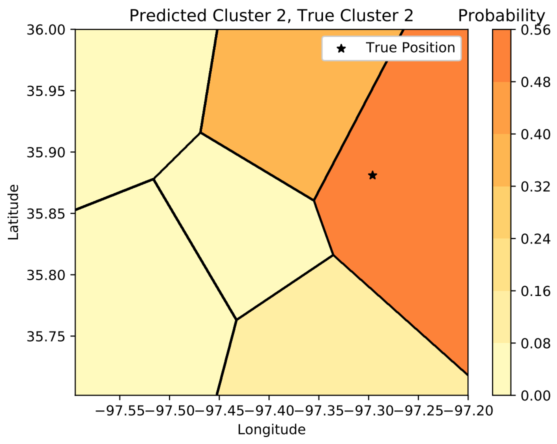
(a) Correct Classification of Cluster 2



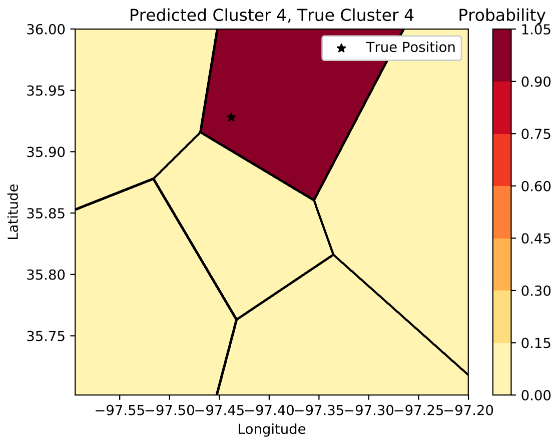
(b) False Classification



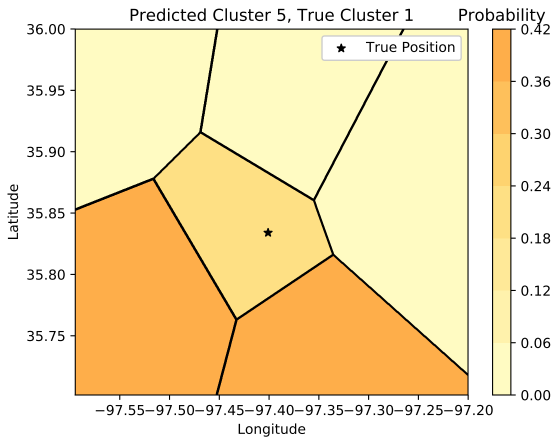
(c) Correct Classification of Cluster 5



(d) Correct Classification of Cluster 2



(e) Correct Classification of Cluster 4



(f) False Classification

Figure 9: Different test samples visualized as probabilistic map on the domain of interest. Some of the samples are correctly predicted, some samples are misclassified. The plot were generated using the code of [3] and using the proposed CNN-LSTM architecture.

Appendix F Class Diagram

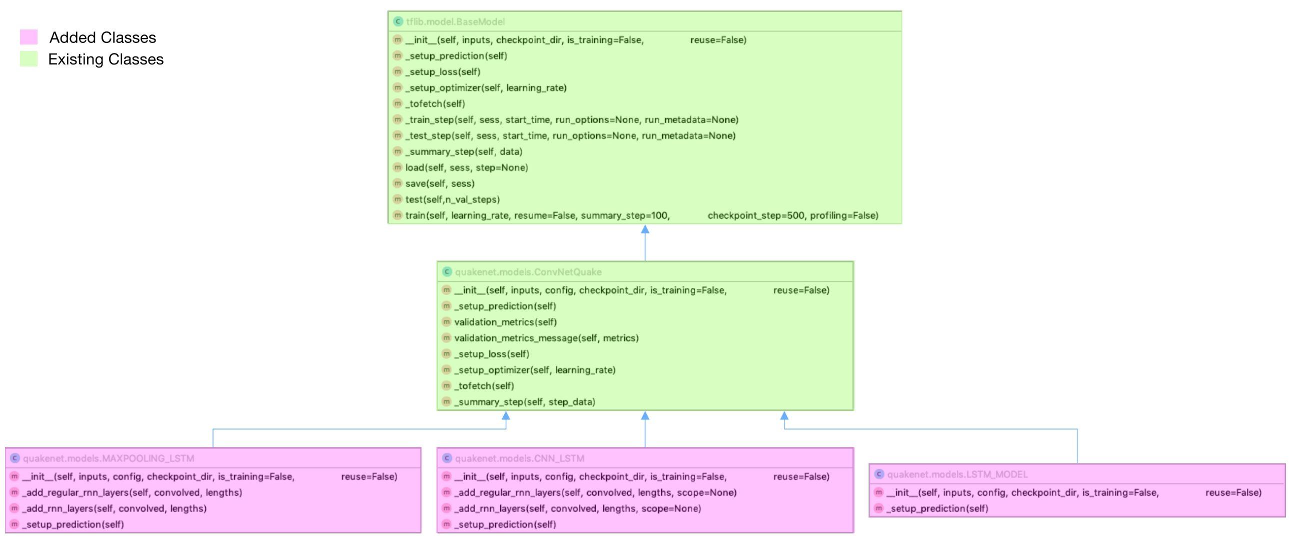

Added Classes
Existing Classes
tflib.model.BaseModel
__init__(self, inputs, checkpoint_dir, is_training=False, reuse=False)
_setup_prediction(self)
_setup_loss(self)
_setup_optimizer(self, learning_rate)
_tofetch(self)
_train_step(self, sess, start_time, run_options=None, run_metadata=None)
_test_step(self, sess, start_time, run_options=None, run_metadata=None)
_summary_step(self, data)
load(self, sess, step=None)
save(self, sess)
test(self,n_val_steps)
train(self, learning_rate, resume=False, summary_step=100, checkpoint_step=500, profiling=False)
quakenet.models.ConvNetQuake
__init__(self, inputs, config, checkpoint_dir, is_training=False, reuse=False)
_setup_prediction(self)
validation_metrics(self)
validation_metrics_message(self, metrics)
_setup_loss(self)
_setup_optimizer(self, learning_rate)
_tofetch(self)
_summary_step(self, step_data)
quakenet.models.MAXPOOLING_LSTM
__init__(self, inputs, config, checkpoint_dir, is_training=False, reuse=False)
_add_regular_rnn_layers(self, convolved, lengths)
_add_rnn_layers(self, convolved, lengths)
_setup_prediction(self)
quakenet.models.CNN_LSTM
__init__(self, inputs, config, checkpoint_dir, is_training=False, reuse=False)
_add_regular_rnn_layers(self, convolved, lengths, scope=None)
_add_rnn_layers(self, convolved, lengths, scope=None)
_setup_prediction(self)
quakenet.models.LSTM_MODEL
__init__(self, inputs, config, checkpoint_dir, is_training=False, reuse=False)
_setup_prediction(self)

Figure 10: Class Diagram of the Model inheritance structure. The new model classes *Maxpooling LSTM*, *CNN LSTM* and *LSTM* are child classes of the original *ConvNetQuake* class. The new classes deploy class functions that overwrite the `_setup_prediction` function in order to build the neural network for each specific architecture.

Appendix G Evaluation and Training Logs from the Cluster

The evaluation of 243000 windows à 10 seconds each took on the compute cluster about 3 minutes and 20 seconds. The raw test output of this computation can be found in Listing ?? and furthermore this function exports a `csv` file which contains the classified clusters. The test set was constant for all test cases and it consists of the continuous waveform of the month July 2014.

Listing 1: ConvNetQuake Evaluation Log File

```
python bin/predict_from_tfrecords.py --dataset data/tfrecords_continuous_detection/July2014 --
    checkpoint_dir models/convnetquake --n_clusters 6 --max_windows 2678400 --output output/
    july_detections
GSOK029_7-2014.tfrecords
Catalog created to store events output/july_detections/catalog_detection.csv
Loaded model at step 32000 from snapshot models/convnetquake/model-32000.
Predicting using model at step 32000
processed 1000 windows
processed 2000 windows
...
...
...
processed 242000 windows
processed 243000 windows
Evaluation completed (1 epochs).
joining data threads
Prediction took 3.0 min 19.4743001461 seconds
```

Listing 2: ConvNetQuake Evaluation Log File

```
Loaded model at step 19500 from snapshot ../output/training_till/ConvNetQuake/model-19500.
Evaluating at step 19500
128 | loss = 1.10170 | det. acc. = 100.0% | loc. acc. = 76.6%
256 | loss = 0.99092 | det. acc. = 100.0% | loc. acc. = 79.7%
...
...
...
5120 | loss = 1.08875 | det. acc. = 100.0% | loc. acc. = 73.4%
5248 | loss = 1.06865 | det. acc. = 100.0% | loc. acc. = 75.0%
Evaluation completed (50 epochs).
5248 windows seen
Average | loss = 1.07361 | det. acc. = 100.0% | loc. acc. = 75.3%
```

Listing 3: CNN LSTM Training Log File

```
USING CNN LSTM MODEL
Initializing all variables.
Starting data threads coordinator.
Starting optimization.
Step 10 | 13s (1312ms) | loss = 1.4201 | det. acc. = 64.1% | loc. acc. = 54.7%
Step 20 | 24s (1211ms) | loss = 1.4103 | det. acc. = 64.1% | loc. acc. = 55.5%
Step 30 | 35s (1173ms) | loss = 1.4068 | det. acc. = 69.5% | loc. acc. = 51.6%
Step 40 | 46s (1158ms) | loss = 1.3325 | det. acc. = 68.8% | loc. acc. = 56.2%
...
...
...
Step 7450 | 7762s (1042ms) | loss = 0.4594 | det. acc. = 99.2% | loc. acc. = 87.5%
Step 7460 | 7773s (1042ms) | loss = 0.5494 | det. acc. = 98.4% | loc. acc. = 84.4%
Step 7470 | 7784s (1042ms) | loss = 0.5546 | det. acc. = 99.2% | loc. acc. = 81.2%
Step 7480 | 7794s (1042ms) | loss = 0.5514 | det. acc. = 100.0% | loc. acc. = 83.6%
Training completed at step 7484.
Shutting down data threads.
Waiting for all threads.
Optimization done.
```

Listing 4: CNN LSTM Evaluation Log File

```
Loaded model at step 5120 from snapshot ../output/training_till/CNN_LSTM_130_00_64/model-5120.
Evaluating at step 5120
128 | loss = 0.76979 | det. acc. = 100.0% | loc. acc. = 83.6%
256 | loss = 0.75857 | det. acc. = 100.0% | loc. acc. = 85.9%
384 | loss = 0.88161 | det. acc. = 100.0% | loc. acc. = 80.5%
512 | loss = 0.85831 | det. acc. = 100.0% | loc. acc. = 80.5%
...
...
...
```

```
10112 | loss = 0.75992 | det. acc. = 100.0% | loc. acc. = 85.2%
10240 | loss = 0.79140 | det. acc. = 100.0% | loc. acc. = 85.2%
10368 | loss = 0.86955 | det. acc. = 100.0% | loc. acc. = 79.7%
10496 | loss = 0.84655 | det. acc. = 100.0% | loc. acc. = 82.0%
Evaluation completed (100 epochs).
10496 windows seen
Average | loss = 0.81475 | det. acc. = 100.0% | loc. acc. = 82.9%
```

References

[1] National Research Institute for Earth Science and Disaster Resilience. High-Sensitivity Seismograph Network. `http://www.hinet.bosai.go.jp/`, 2018. [Online; accessed 14-Oktober-2018].

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] Thibaut Perol, Michaël Gharbi, and Marine Denolle. ConvNetQuake. `https://github.com/tperol/ConvNetQuake`, 2016. [Online; accessed 29-Oktober-2018].

[4] Thibaut Perol, Michaël Gharbi, and Marine Denolle. Convolutional neural network for earthquake detection and location. *Science Advances*, 4(2), 2018.

[5] Tensorflow. Recurrent Neural Networks for Drawing Classification. `https://www.tensorflow.org/tutorials/sequences/recurrent_quickdraw`, 2018. [Online; accessed 22-November-2018].