Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

AF Classification from a short single lead ECG recording: the PhysioNet — Computing in Cardiology Challenge 2017

Csaba Botos

2017

A review submitted in partial satisfaction of the requirements of Individual laboratory

Advisor: PhD. István Z. Reguly

Abstract

We have developed an algorithm combining machine learning and handcrafted features with medical relevances. Our architecture consists of three main parts:

Deep Learning. using a convolutional neural network for initial feature extraction, preserving the temporal aspects of the data. These time dependent features are fed into the recurrent network which handles the temporal aspects of the data and generates a feature vector with fixed length.

QRS complex filtering. Another set of features is based on the Pan Tompkins algorithm to identify the characteristic wave descriptors (QRS complexes) of the data. The variance of the detected wave complex positions along the variability index of P and T waves, which describe the morphology of the whole cardiac cycle, are also used in the classification.

Frequency domain analysis. The last set of handcrafted features uses Fourier descriptors of the data. The Fourier domain of the training data was analyzed and those frequencies which show a large difference between normal and fibrillation cases were selected. The biological relevance of the selected frequencies was examined and advised by cardiologists.

In the current implementation we concatenate the feature vectors coming from the three previously described parts and these features are classified by a Multi Layer Perceptron.

Keywords: ***deep learning, residual network, fully convolutional network, time series, recurrent network, transfer learning***

Contents

1. Introduction

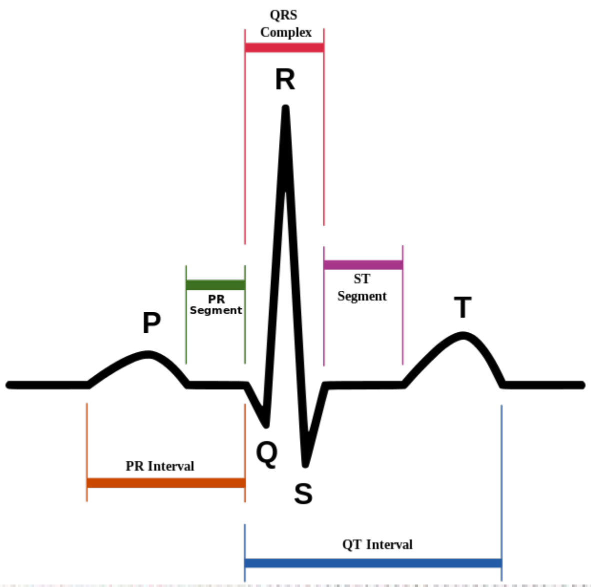
Social relevance

Atrial Fibrillation is the most common type of cardiac arrhythmia. Thousands of heart failures could be prevented by proper treatment if early signs were diagnosed in time. Over the years medical equipment advanced by involving some sort of artificial intelligence. We don't have to go far, probably every gas station in the area has a semi-automatic defibrillator, that has a sensor built in which recognizes whether the patient needs to be shocked or not — to prevent unnecessary reanimation. The basic idea is to relieve overwhelmed doctors by recognizing invariant patterns in the specific cases. These patterns are taught in medical universities, and fine-tuned during years of practice — but turns out that volunteering cardiologists can submit their knowledge base to engineers who in return will automatize at least the trivial process to support the better treatment. In order to assist in the diagnosis, and make predictions based on previous cases we utilize Machine Learning techniques to combine professional knowledge and neural networks to achieve the lowest error rate on a classifying task.

The real challenge

This year the PhysioNet Challenge [1] is simple, a set of single lead ECG samples labeled by a team of cardiologists is provided. The labels are the following: *noisy, normal, atrium fibrillation, other*. The task is to develop a method that is able to classify from an unseen prerecorded sample. At first sight the initial conditions are very encouraging, but soon after the first check of the training files it turns out that only a limited number of samples are given, exactly 8528 samples, the dataset has a strongly imbalanced distribution between classes.

Also as it was revealed during the last weeks, the ground truth had several errors, misleading the training process. We had several discussions in the early phase of development with professionals about possible classification errors, that even were trivial to us 1.2. Many suggestions were committed by the competitors, and as a result the organizers released an



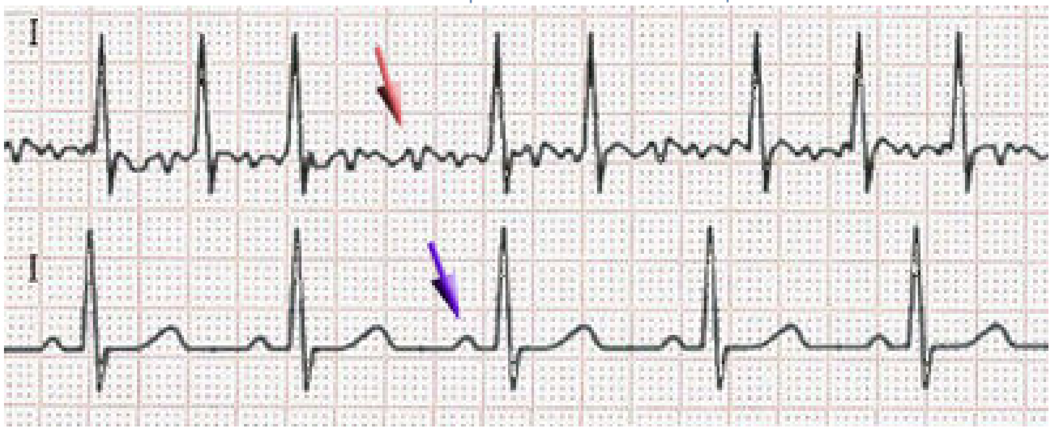


Figure 1.1: Top: QRS waves, a special type of waves complex that can be observed during a healthy functional cardiac cycle, sinus rhythm. Bottom: academic example of the normal sinus rhythm (lower, purple arrow points to P wave) compared to an ECG of atrial fibrillation (upper, red arrow points to the missing P wave), lacking the P waves. Image credits: Wikimedia.

updated label reference. The difference depicted in details on Figure 1.3.

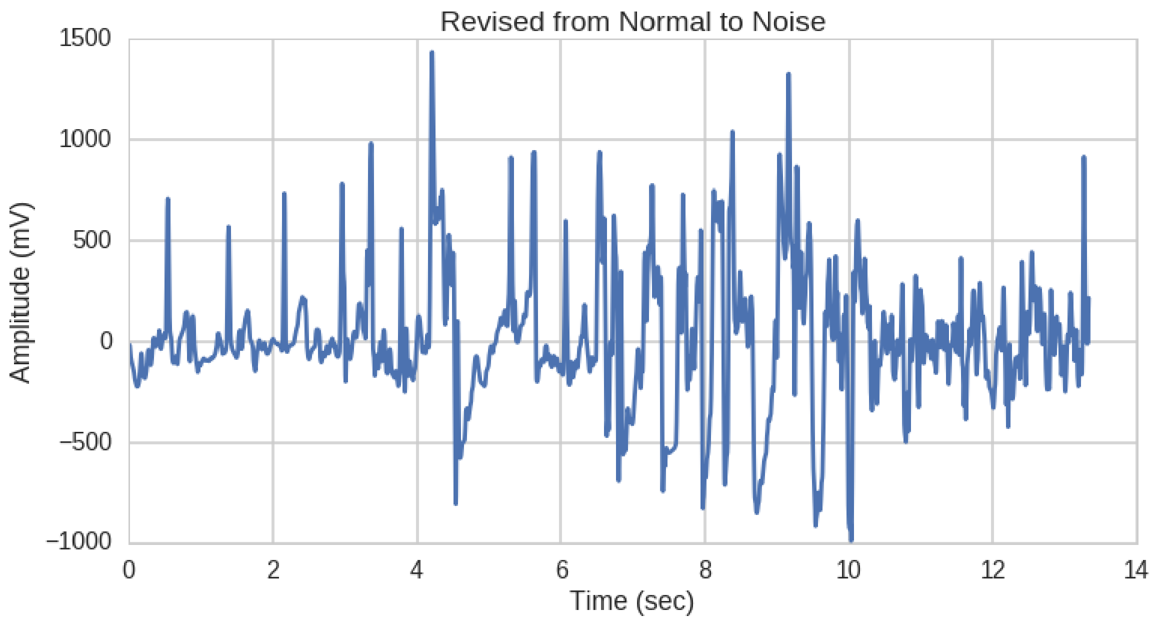


Figure 1.2: The depicted ECG signal was originally classified as a *normal* sample, later revised to *noisy*.

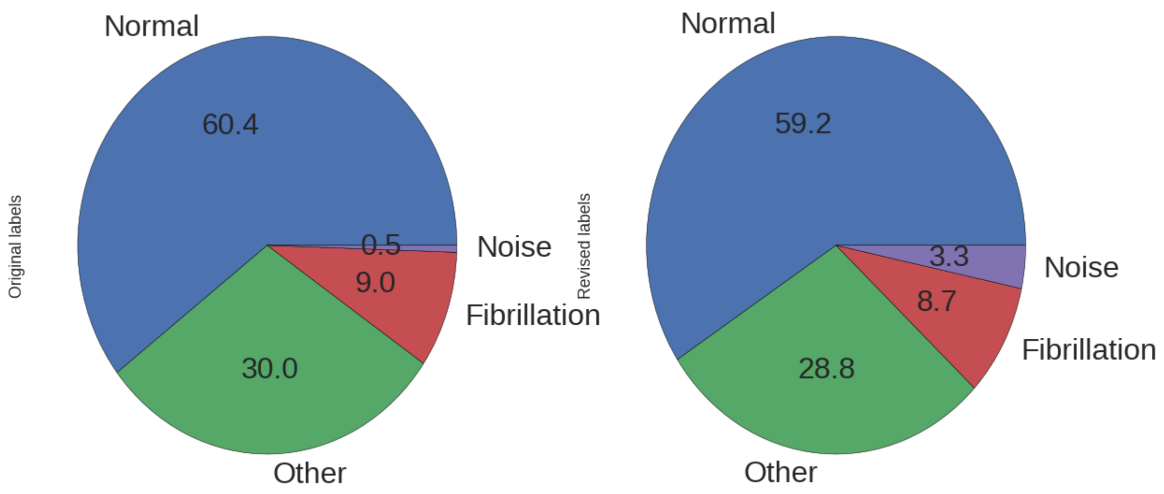


Figure 1.3: The differences between the original (left) and the revised (right) ground truth labels. According to the latest update, the noise class is still underrepresented, and still it is very likely that any model trying to discriminate noisy samples will be too simple to classify them all correctly, or with larger capacity will be over-fitted.

2. Related Works

Motivation. "Previous studies concerning AF classification are generally limited in applicability because 1) only classification of normal and AF rhythms were performed, 2) good performance was shown on carefully-selected often clean data, 3) a separate out of sample test dataset was not used, or 4) only a small number of patients were used. It is challenging to reliably detect AF from a single short lead of ECG, and the broad taxonomy of rhythms makes this particularly difficult. In particular, many non-AF rhythms exhibit irregular RR intervals that may be similar to AF. In this Challenge, we treat all non-AF abnormal rhythms as a single class and require the Challenge entrant to classify the rhythms as 1) Normal sinus rhythm, 2) AF, 3) Other rhythm, or 4) Too noisy to classify." — AF Challenge 2017 introduction

Suggested features recognized by cardiologists [23]. Published methods for analysis of absence of P waves or the presence of fibrillatory f waves in the TQ interval: Echo state network [21], P-wave absence (PWA) based detection [13], analysis of the average number of f waves [4], P-wave-based insertable cardiac monitor application [22], wavelet entropy [2, 24] and wavelet energy [6], Pan Tompkins QRS detection [31].

Guideline on combining Machine Learning techniques with hand crafted features [7]. Deep Learning for time series classification [14, 30].

3. Models

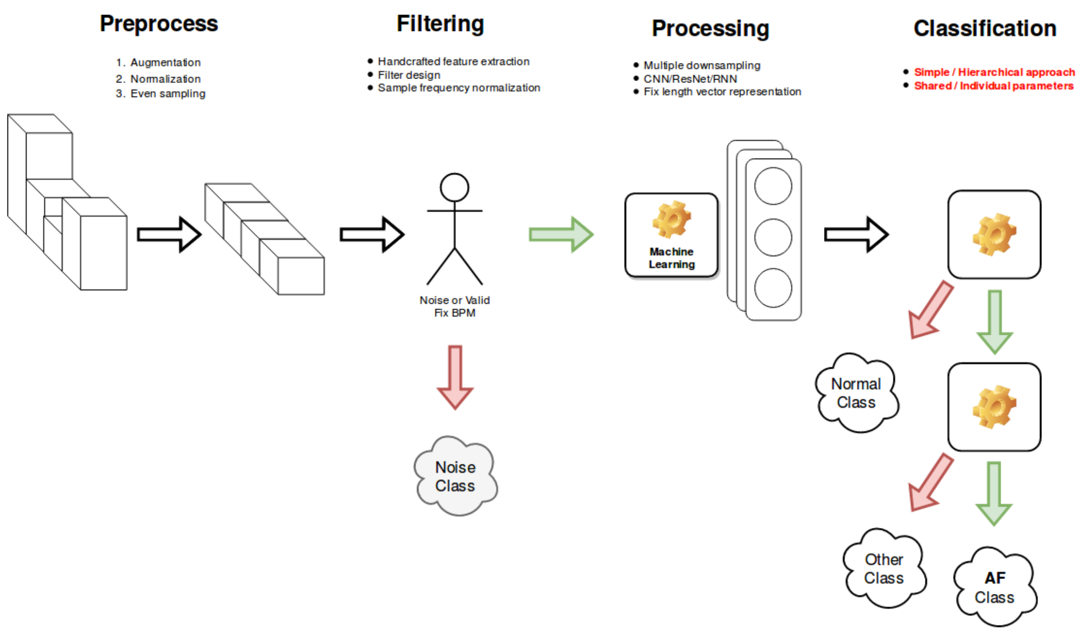
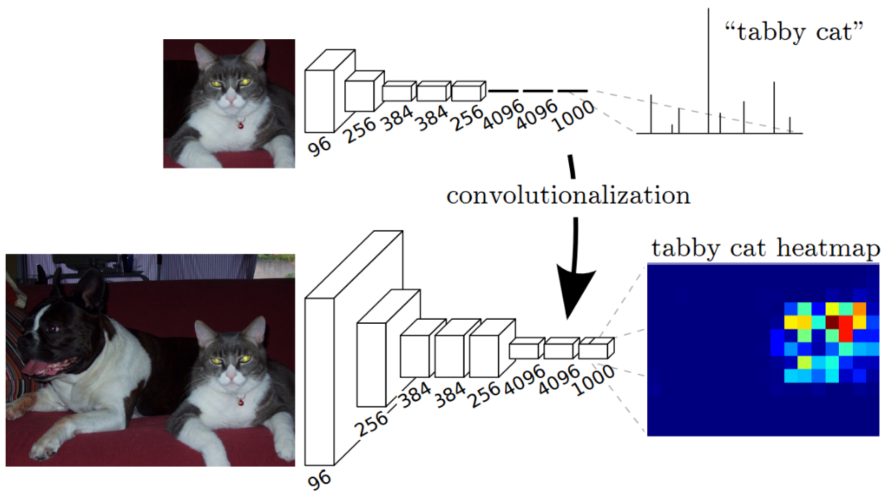


Figure 3.1: We believe that our best chance to classify samples properly is that we learn how to separate *valid* and *noisy* data from one-another — since the model will not be forced then to be able to recognize delicate patterns which could indicate heart failure along with harsh noise patterns. In order to avoid overfitting on the small train set of a few hundred samples, instead simply training a binary classifier network, we are working hard to filter these noisy samples by other handcrafted approaches. Next, the sample should be categorized as a healthy or an unhealthy ECG. As a final step, we are training a network to distinguish atrium fibrillation from other ill ECG. During the training process the different part of the model are trained separately only on their own domain, and after descending below a certain error rate, we assemble the model.

As mentioned before, we intended to exploit the recent success of neural networks that offer plenty of methods and architectures — and a few rules of thumb to keep in mind when preparing the blueprint. Currently we are experimenting with hierarchical classification models, for a quick overview see Figure 3.1. At the time of writing I will describe the main building blocks we have tried so far, and present the baseline our recent tests are based upon. My role in the group is to implement and maintain the deep learning back-end, also



Source: http://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Figure 3.2: Semantic segmentation (bottom) uses the feature extractors from VGG (top), by simply reshaping the fully connected layers into convolutions with local receptive fields. We expect the same behavior when training convolutional networks on one dimensional time series. Image credits [16]

to integrate the hand crafted features, and carry out experiments and evaluate them to determine which combination leads to the most successful architecture.

Variable length representation

First of all, we have to deal with time dependency. Since we cannot determine how long the samples will be, nor describe a time frame that would fit every relevant pattern, we have to project somehow the sample into a fixed dimensional space. In order to preserve important information by the projection we can separate samples into clusters by previously extracting features which describes best the data. These values are usually time dependent pattern fitting maps.

Fully Convolutional Networks. Based on the fact that hierarchical feature extractors such as the LeNet [15], and the VGG [26] are easily trained, reliable, and can be easily restructured for transfer learning purposes. I decided to re-implement the Fully Convolutional Network [16] paradigm in one dimension, as motivated by the following works [19, 14]. The actual convolutional layers of our FCN models are following the pattern:

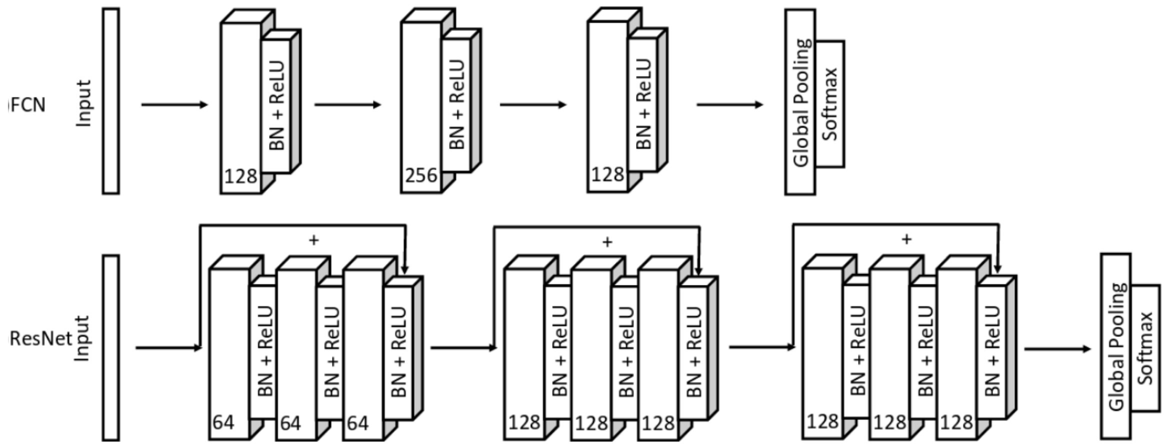


Figure 3.3: Image credits [30]

$$
\begin{aligned}
c &= \mathbf{K} * x \\
b &= BatchNorm(c) \\
h &= ReLU(b) \\
y &= AvgPool_w(h)
\end{aligned}
$$

Where BatchNorm is described in [11], and ReLU is the rectified linear unit (nonlinear) activation function, and w indicates the window size of the pooling operation (the stride was set to the same value as well). Hyper-parameters of this convolutional layer are hidden in the dimensions of $\mathbf{K} \rightarrow$ `[IN, OUT, k]`, where `IN` is fixed, `OUT` represents how many feature maps will be created by the convolution operation, and `k` is how large the kernel will be, i.e. how large the receptive field will this layer have on x. These layers were assembled in the depicted manner on Figure 3.3 (top row).

Residual Networks. We also began experimenting with Deep Residual Networks [9], suggested by [30]. The basic idea is simple: we concatenate several of the above mentioned FCNs after each other, while stabilizing the gradient flow by adding the convolutional blocks' input to their output. However the interpretations of why deep residual networks actually work so well are not clear. An excellent article on this topic van be found at[27]. Oversimplifying, we could say each FCN blocks (in Eq. 1) in the model are no longer responsible for detecting every feature, but expected as shallow networks to work together as different ensembles, which allows us to increase the overall capacity of the model.

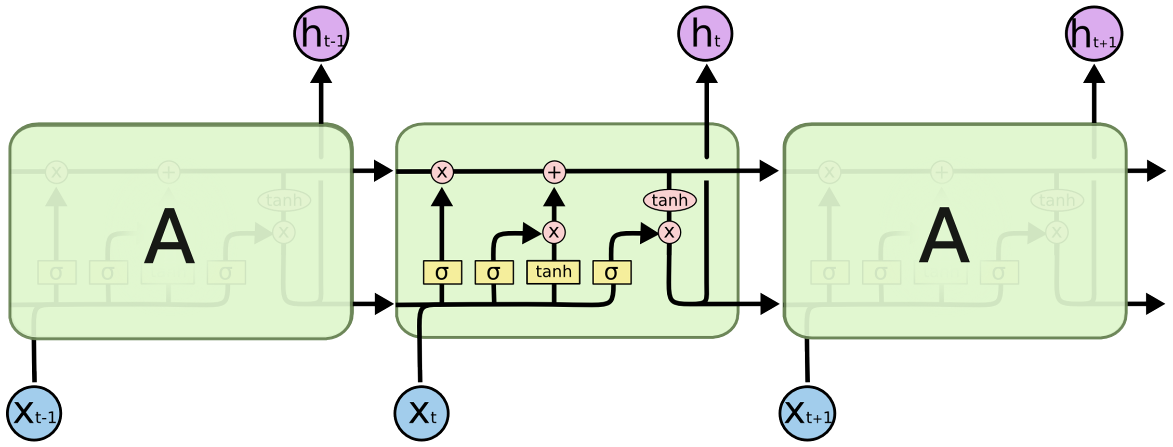


Figure 3.4: Long Short Term Memory network maintains two state, one hidden, and one output state, and via forget and input gates it offers a solution for training recurrent neural networks with back-propagation through time without exploding/vanishing gradients. Image credit: Chris Olah's blog [20]

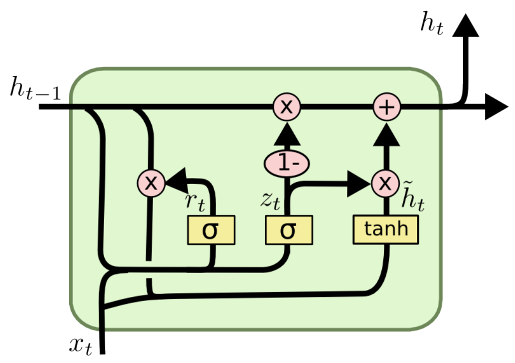
Fixed length representation

Once every filters have been evaluated in the end we can use different approaches: to take the weighted norm or mean of the filtered signal to represent each pattern's presence in the input signal or we can use techniques that have been proven to be successful in natural language processing tasks: recurrent neural networks (RNNs). Our first attempt was to use a Long Short Term Memory network (LSTM) [10, 17], shortly we switched to its modified version Gated Recurrent Unit network (GRU) [3], but we experienced that the network was not capable to overfit the train set, and achieve a satisfactory error rate.

Due to the large number of task specific hyper-parameters in LSTMs and GRUs the search space would exponentially increase, so we decided to continue experimenting with simply reducing the feature map to a single scalar value by mean averaging the activation through time of the feature extractors.

Augmentation and transfer learning

Despite the initial difficulties with RNN, I began to experiment as a spinoff project to apply RNN as a time series predictor, training it to predict the following N values of the measurements based on the previous parts of the sample. I believe that this model can be reused to initialize training classifier recurrent networks, instead of random initialization. On the other hand, by feeding back the network its own predictions we could produce data 3.6. The network in this case would function as some sort of "language model" [18] of ECG signals - and if trained properly, i.e. on a single class, it would output labeled, yet totally artificial



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 3.5: Gated Recurrent Unit networks combine forget and input gates into a single update gate. By discarding the output cell state, the result is a simpler model, that is easier to initialize, and train. Image credit: Chris Olah's blog [20]

samples, which could be used as augmented training data as well.

Classification

After extracting the most relevant time features from the sample, and reducing it to a fix length feature vector, the model has to separate the feature space with respect to the classes. A basic approach is to apply logistic regression where each feature dimension is taken into account and weighted to determine each class (by dissecting the feature space with hyper-planes). As a result a four dimensional vector $[0,1]^4$ is produced, where each dimension represents the confidence of the network whether the sample is belonging to the corresponding class or not. The output is then normalized with the softmax function to give a confidence distribution over the classes:

$$p_i = \frac{\exp(y_i)}{|\exp(y)|_1} \tag{3.1}$$

We also tried applying Multiple Layer Perceptrons (MLP) [8], on top of these features, however the results show that the model is less likely to collapse during training due to falling in a local minimum.

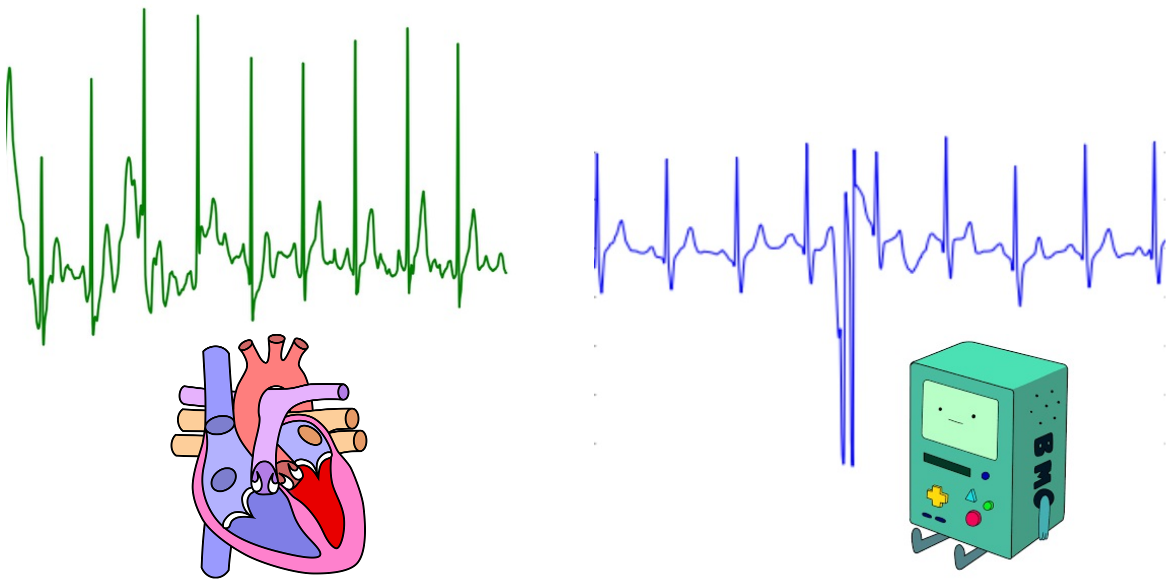


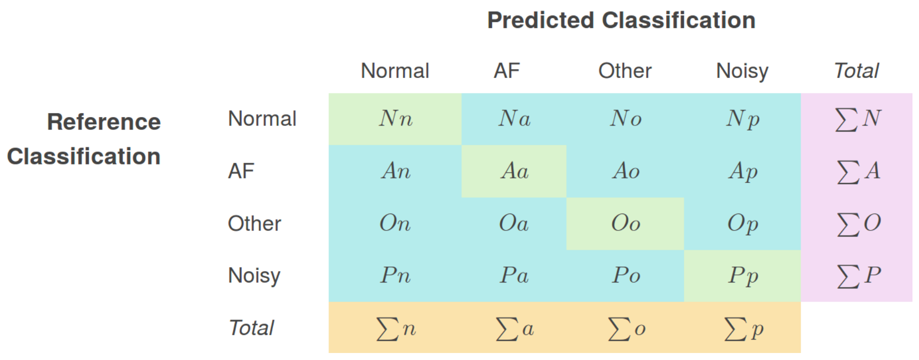
Figure 3.6: Artificial sample produced by an LSTM trained to predict and continue original ECG samples. Left: an AF Challenge 2017 entry classified as *normal*. Right: ECG sequence produced by a network trained on exclusively *normal* samples. In this case the network functions as some sort of language model of ECG. Among many samples, professionals could not tell whether samples produced by this method were artificial or taken from the original training set. Truth on being told, after further analysis they could detect artifacts, still some said that a very ill patient could produce the same samples. Image credits: Wikimedia, Adventure Time.

4. Training and evaluation

The best practice on evaluating the classifier model's performance is to simply compare the number of matching labels evaluated on a completely disjunct set of samples from the training set. However in our case it was not so trivial to score different methods, since by simply answering *normal* to every sample would yield 50+% success rate, therefore it would be less representative.

Confusion matrix

Instead we are using an extended version of precision-recall evaluation, namely we apply a *confusion matrix*, where every row represents a histogram of the correct label, and every column represents the predictions of our model. The diagonal elements show how many labels match, but it preserves the distribution between classes. Using this method, we can easily track down when a training routine collapses, and the network is only using a few of the available classes. When the diagonal elements over number the off-diagonals, then the network has been succesfully trained. For examples of the confusion matrix see Figure **??**. By reducing the matrix to a single scalar defined on the website of AF Challenge, we get an accuracy value, which tells us the exact score we would obtain by submitting an official entry on behalf of our research group.



Predicted Classification

Reference Classification

	Normal	AF	Other	Noisy	*Total*
Normal	Nn	Na	No	Np	$\sum N$
AF	An	Aa	Ao	Ap	$\sum A$
Other	On	Oa	Oo	Op	$\sum O$
Noisy	Pn	Pa	Po	Pp	$\sum P$
Total	$\sum n$	$\sum a$	$\sum o$	$\sum p$	

Figure 4.1: Confusion matrix. Diagonal values represent prediction **hit**, off-diagonals represent **miss**. Image credits AF Challenge 2017

The training set

In order to keep track of how well our model generalized the features, and to avoid over-fitting, the original downloaded samples were separated into three sets. First we selected 20% of every class to keep it as a **test** set. As a next step, the remaining 80% were fed to an augmentation algorithm that added additive and multiplicative noise sampled from normal distribution biased to match the moment of each individual sample, and re-sampled each class' randomly selected entries to make an even distribution between classes. Finally the augmented data was separated into **train** and **evaluation** set in an 80-20 ratio. We use evaluation set to keep track how our model performs on unseen data. Different *training policies* modify learning rate, perturb the network parameters, or early stopping when over-fit occurs for given number of consecutive iterations.

Data standardization

We experimented with different normalization techniques such as standard normalizing per sample, or with the global mean and average with no significant difference in the results Our current experiments involve standardizing the samples by heart rate frequency, thus the class relevant invariance can be easier learned by the the network.

Toy problem. A simple example is the following: suppose we have two patients, one who drinks coffee regularly resulting in a high heart rate (even in healthy) and one who runs marathon every week with low heart rate at rest. If both were having the same cardiac symptom probably the cardiologist would record their natural heart rate to have a baseline, a reference point to use for diagnosis. This is because the time domain patterns are expressively specific to the patient's heart rate. Technically this means, if the atrial fibrillation specific curve would appear in both samples it would probably appear elongated or compressed — and the model had to learn both patterns as if no similarity would have exist between them. This problem could lead to a case where the network would discard some rare feature, in order to reserve the capacity for the trivial pattern just with different lengths.

Solutions. To counterweight we could increase the capacity of the network, which often leads to significant improvement in performance on the train set, however at the same time as a by-product the performance on validation set decays, since the model is more likely to over-fit, simply remember the train samples, and loses its ability to generalize well. The loss is likely to happen since our training set only contains less than 8K samples. An other solution is standardizing the heart rate frequency of our samples by resizing the whole sequence in time, which would in case, solve our toy example. We made sure that discarding the sample specific frequency has no significant relevance to the corresponding class. For details of class BPM variance see Figure 4.2

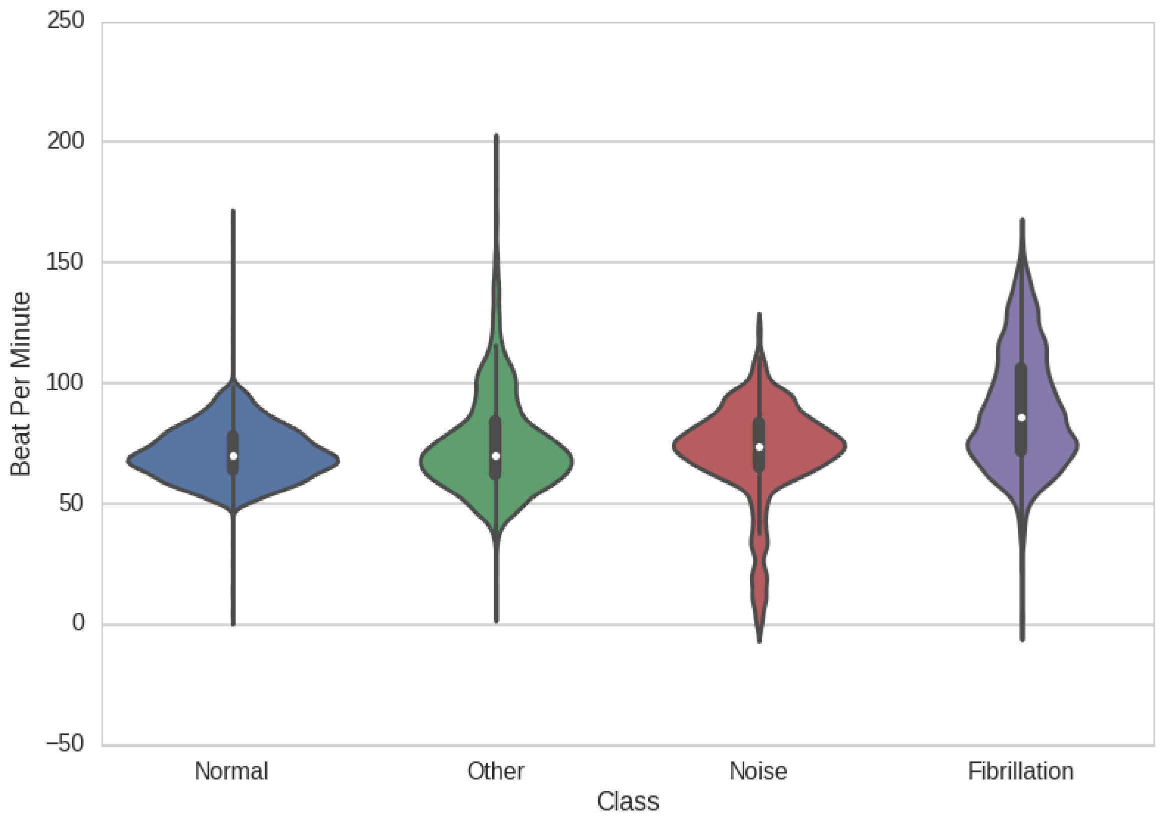


Figure 4.2: The heart rate distribution and variance between classes is similar (except of cases in Noise class where the heart beat detection algorithm fails to find the R waves, thus resulting in extreme values, which we intend to filter before the entry reaches the valid sample classifiers), so discarding the average heart rate by normalizing every sample by the value, presumably would not affect negatively the performance of our network.

Regularization

Weighted loss. Our first attempts to improve accuracy on rare samples was to set higher loss on underrepresented classes to encourage the network to optimize by learning to recognize these samples better, but it resulted in over-fitting and soon was replaced by augmented evenly distributed mini-batches. *Mini-batch* is a common method throughout almost every Deep Learning algorithm. It enables the back-end to train the network with multiple samples at once. Both parallel inference and weight adjusting is implemented in the latest machine learning frameworks.

Default optimizer, and other regularization. We are using the current state-of-the art optimizer algorithm ADAM [12] and DropConnect [29] with $p = 0.5$ settings. We also apply weight decay, by adding L_2 norm of each θ in the networks parameters to the overall loss function and use soft labels (perturbed one-hot vectors) in order to prevent the networks from favoring a specific class. Our training policy varies over different settings, currently we are using early stopping method: after 10 consecutive steps where validation performance have not been improved the trainer shuts down, to enable other train routines use the allocated device — otherwise the models are evaluated after a previously given number of steps. Throughout the training, we decrease exponentially the learning rate, in order to prevent the model from oscillation.

5. Results

Here we list the evaluation scores, and further guides we use for planning our next steps, available at the time of writing.

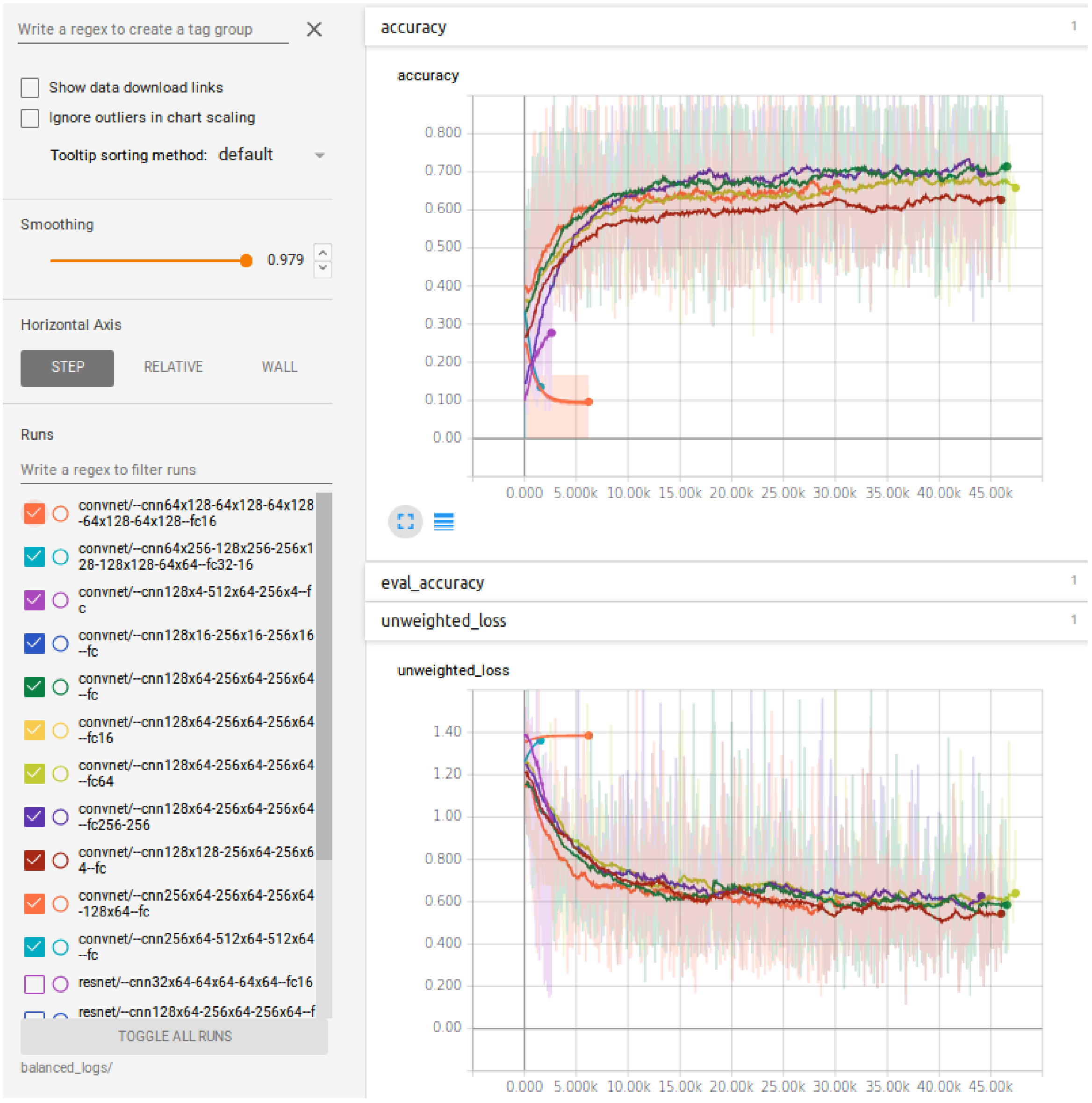
Fully Convolutional Network

Our overall impression with convolutional networks is that, they are slowly training, even high capacity networks are unable to achieve satisfactory score on training set 5.1. For this reason, we aimed first to use an architecture, that is capable to over-fit on the train data, so we could fine-tune the training by enforcing the regularization settings.

MLP problem. A strange result is that by adding MLP block before the classifier, the performance was drastically reduced, and soon the training collapsed. The models were biased towards only choosing a single class. Excellent example for this behavior can be seen on Figure 5.1, where models with MLP built in oscillate around the same performance throughout an *epoch*. (epoch: a set of training steps which includes every entry from the training set.) We can also check out the confusion matrix of these models on Figure 5.3, also the histogram of the model parameters reveals, that the network is strongly biased towards choosing one specific class independently from its input in Figure 5.2.

Residual Network

Our next choice were deep residual networks. With this model, we could achieve better training and evaluation scores. The exclusion of the MLP block had to be made when using ResNets also, because the same behavior occurred when we applied hidden layers between feature extractors and the classifier layer.


Write a regex to create a tag group
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.979
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
TOGGLE ALL RUNS
balanced_logs/
accuracy
accuracy
0.800
0.700
0.600
0.500
0.400
0.300
0.200
0.100
0.00
0.000 5.000k 10.00k 15.00k 20.00k 25.00k 30.00k 35.00k 40.00k 45.00k
eval_accuracy
unweighted_loss
unweighted_loss
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
0.000 5.000k 10.00k 15.00k 20.00k 25.00k 30.00k 35.00k 40.00k 45.00k

☐ Show data download links
☐ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing: 0.979
Horizontal Axis: [✓ STEP] RELATIVE WALL

Runs
- ✓ convnet/--cnn64x128-64x128-64x128-64x128-64x128--fc16
- ✓ convnet/--cnn64x256-128x256-256x128-128x128-64x64--fc32-16
- ✓ convnet/--cnn128x4-512x64-256x4--fc
- ✓ convnet/--cnn128x16-256x16-256x16--fc
- ✓ convnet/--cnn128x64-256x64-256x64--fc
- ✓ convnet/--cnn128x64-256x64-256x64--fc16
- ✓ convnet/--cnn128x64-256x64-256x64--fc64
- ✓ convnet/--cnn128x64-256x64-256x64--fc256-256
- ✓ convnet/--cnn128x128-256x64-256x64--fc
- ✓ convnet/--cnn256x64-256x64-256x64-128x64--fc
- ✓ convnet/--cnn256x64-512x64-512x64--fc
- ☐ resnet/--cnn32x64-64x64-64x64--fc16
- ☐ resnet/--cnn128x64-256x64-256x64--f

Figure 5.1: Overall performance of the FCN approaches plotted in TensorBoard. Top: Accuracy derived from the confusion operator. Bottom: Unweighted loss during training

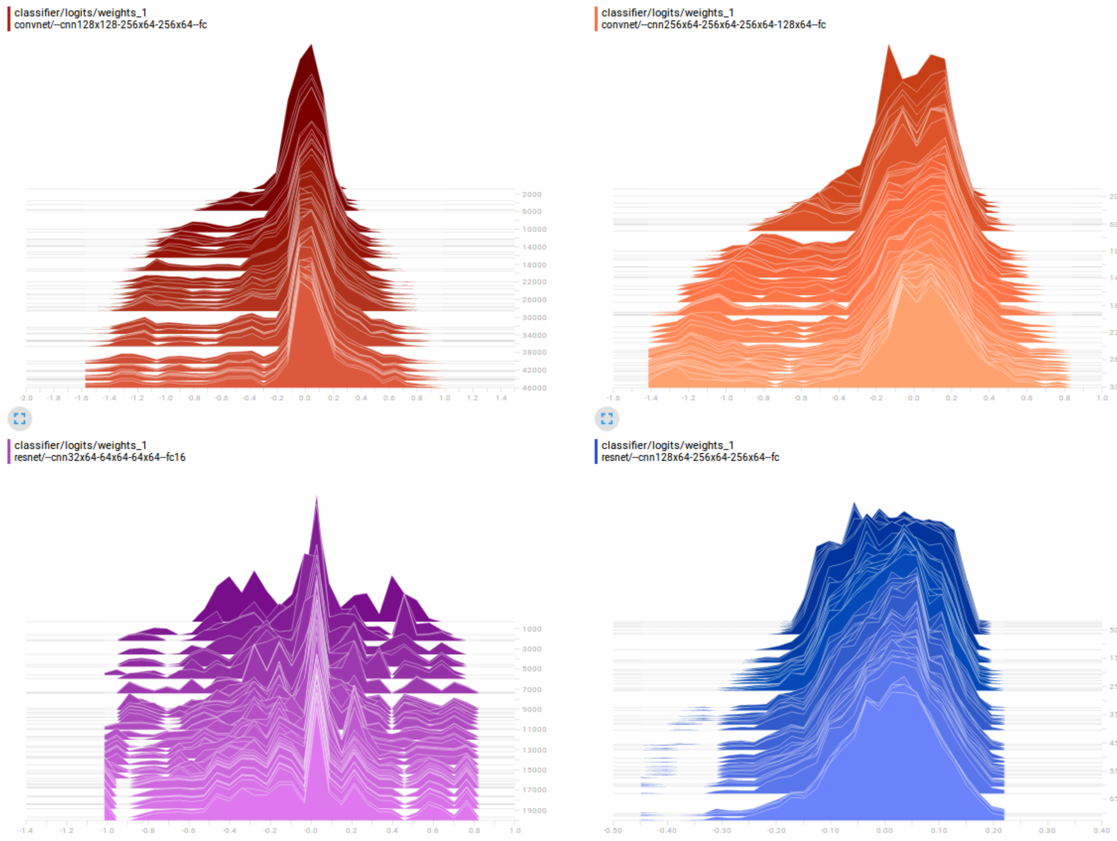


Figure 5.2: Weight distribution of the classifier (last) layer through training. Note that networks with small variance, and local edgy peaks are more likely to fall into a local minimum of the parameter space — where choosing the same class no matter what is the input is too stable state for the network to learn any further features.

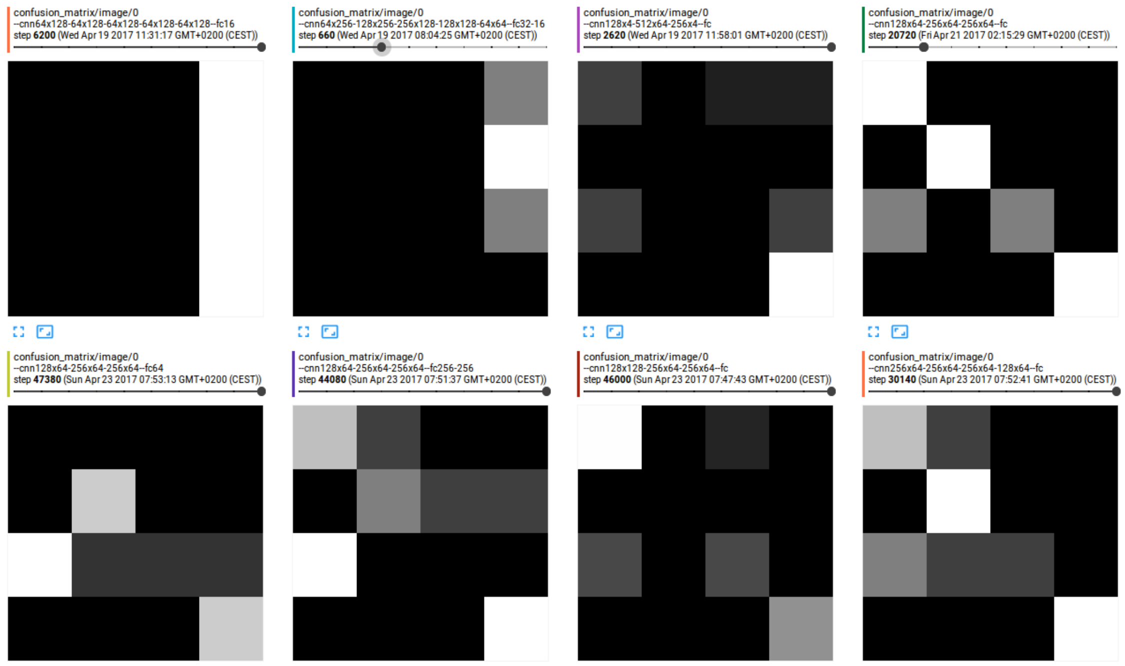
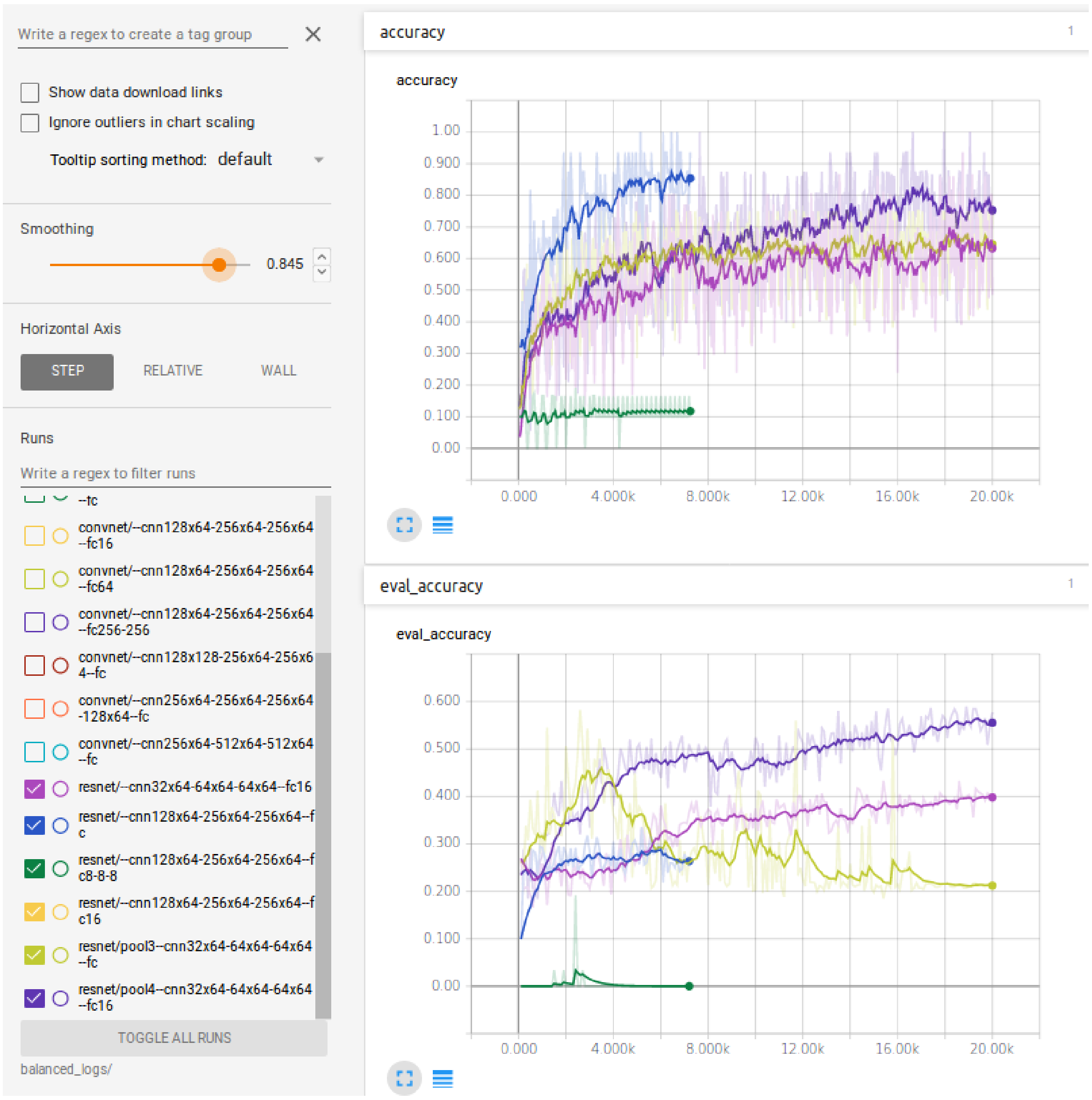

confusion_matrix/image/0
--cnn64x128-64x128-64x128-64x128-64x128--fc16
step **6200** (Wed Apr 19 2017 11:31:17 GMT+0200 (CEST))
confusion_matrix/image/0
--cnn64x256-128x256-256x128-128x128-64x64--fc32-16
step **660** (Wed Apr 19 2017 08:04:25 GMT+0200 (CEST))
confusion_matrix/image/0
--cnn128x4-512x64-256x4--fc
step **2620** (Wed Apr 19 2017 11:58:01 GMT+0200 (CEST))
confusion_matrix/image/0
--cnn128x64-256x64-256x64--fc
step **20720** (Fri Apr 21 2017 02:15:29 GMT+0200 (CEST))
confusion_matrix/image/0
--cnn128x64-256x64-256x64--fc64
step **47380** (Sun Apr 23 2017 07:53:13 GMT+0200 (CEST))
confusion_matrix/image/0
--cnn128x64-256x64-256x64--fc256-256
step **44080** (Sun Apr 23 2017 07:51:37 GMT+0200 (CEST))
confusion_matrix/image/0
--cnn128x128-256x64-256x64--fc
step **46000** (Sun Apr 23 2017 07:47:43 GMT+0200 (CEST))
confusion_matrix/image/0
--cnn256x64-256x64-256x64-128x64--fc
step **30140** (Sun Apr 23 2017 07:52:41 GMT+0200 (CEST))

Figure 5.3: Models are monitored through the training process: during weight updates a small subset of unseen entries is inferred, and compared to the ground truth labels. In these *confusion matrices* the row sum represents a histogram of the labels, while the column sum would represent the histogram of the network's choice. Their intersection results in the talkative heat maps. Notice that those models whose confusion matrix is mainly diagonal is performing well, and those networks which have been collapsed gives a matrix where only a single column can be seen (i.e. first row, the two rightmost matrix).


Write a regex to create a tag group

☐ Show data download links
☐ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.845

Horizontal Axis: [✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

☐ --fc
☐ convnet/--cnn128x64-256x64-256x64--fc16
☐ convnet/--cnn128x64-256x64-256x64--fc64
☐ convnet/--cnn128x64-256x64-256x64--fc256-256
☐ convnet/--cnn128x128-256x64-256x64--fc
☐ convnet/--cnn256x64-256x64-256x64-128x64--fc
☐ convnet/--cnn256x64-512x64-512x64--fc
☑ resnet/--cnn32x64-64x64-64x64--fc16
☑ resnet/--cnn128x64-256x64-256x64--fc
☑ resnet/--cnn128x64-256x64-256x64--fc8-8-8
☑ resnet/--cnn128x64-256x64-256x64--fc16
☑ resnet/pool3--cnn32x64-64x64-64x64--fc
☑ resnet/pool4--cnn32x64-64x64-64x64--fc16

TOGGLE ALL RUNS

balanced_logs/

Figure 5.4: Overall performance of the ResNet approaches plotted in TensorBoard. It is worth to mention that the network represented by the green curve is using the same hyper-parameters as the network with the highest train performance (blue), still because of the MLP block it fails to learn a generalized representation. Another important feature is that the network represented by the blue line performs way better on the training set, still fails on the evaluation set. This is the classical example of an over-fitted model. Top: Accuracy on the training set. Bottom: Accuracy on the evaluation set.

6. Future

Short-term plans. First, we are testing hierarchical architectures: whether we should share weights between models, train them separately or in one session etc. When the best candidates are found, we continue fine-tuning them by exploring hyper-parameter space.

Our following task will be utilizing One Shot learning [25, 28] to tackle the small-train set problem.

In the meantime, we intend to submit multiple entries trained and selected with different preconditions and different evaluation methods, in order to find out how well our scoring system represents the real performance.

Long-term plans. We would like to implement Domain Adversarial training of Neural Networks (DANN) [5] to transfer our best model's trained parameters to real world applications which may introduce different sample density.

After successfully training our final model, we intend to use the convolutional layers as feature extractors in reverse engineering to find out if we can help cardiologists by providing them ECG patterns our network used as a guideline for classification. For doing so we have multiple choices: we can utilize DeConv nets [33], or apply gradient ascension [32] on the receptive field of perceptrons, or simply take the mean of windows in samples that yields the largest activation in the latent feature representation.

Finally, the device that was used to record the original samples will be commercially available soon, and it is a great opportunity for us to write the inner mechanism of a real life, end-to-end product which is the goal of the AF Challenge 2017.

Bibliography

[1] *AF Classification from a short single lead ECG recording: the PhysioNet/Computing in Cardiology Challenge 2017*. URL: `https://physionet.org/challenge/2017/` (visited on 05/08/2017).

[2] R Alcaraz et al. “Wavelet sample entropy: A new approach to predict termination of atrial fibrillation”. In: *Computers in Cardiology, 2006*. IEEE. 2006, pp. 597–600.

[3] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[4] Xiaochuan Du et al. “A Novel Method for Real-Time Atrial Fibrillation Detection in Electrocardiograms Using Multiple Parameters”. In: *Annals of Noninvasive Electrocardiology* 19.3 (2014), pp. 217–225.

[5] Yaroslav Ganin et al. “Domain-Adversarial Training of Neural Networks”. In: *arXiv:1505.07818 [cs, stat]* (May 2015). arXiv: 1505.07818. URL: `http://arxiv.org/abs/1505.07818` (visited on 05/07/2017).

[6] Manuel García et al. “Application of the relative wavelet energy to heart rate independent detection of atrial fibrillation”. In: *Computer methods and programs in biomedicine* 131 (2016), pp. 157–168.

[7] Pierre Geurts. “Pattern extraction for time series classification”. In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. 2001, pp. 115–127.

[8] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[9] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[10] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[11] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *arXiv:1502.03167 [cs]* (Feb. 2015). arXiv: 1502.03167. URL: `http://arxiv.org/abs/1502.03167` (visited on 05/09/2017).

[12] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). URL: `https://arxiv.org/abs/1412.6980` (visited on 05/08/2017).

[13] Steven Ladavich and Behnaz Ghoraani. “Rate-independent detection of atrial fibrillation by statistical modeling of atrial activity”. In: *Biomedical Signal Processing and Control* 18 (2015), pp. 274–281.

[14] Martin Längkvist, Lars Karlsson, and Amy Loutfi. “A review of unsupervised feature learning and deep learning for time-series modeling”. In: *Pattern Recognition Letters* 42 (2014), pp. 11–24.

[15] Yann LeCun, Yoshua Bengio, et al. “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* June 2015.

[17] Pankaj Malhotra et al. “Long short term memory networks for anomaly detection in time series”. In: *Proceedings.* Presses universitaires de Louvain. 2015, p. 89.

[18] Tomas Mikolov et al. “Recurrent neural network based language model.” In: *Interspeech.* Vol. 2. 2010, p. 3. URL: `http://www.fit.vutbr.cz/research/groups/speech/servite/2010/rnnlm_mikolov.pdf` (visited on 05/09/2017).

[19] Roni Mittelman. “Time-series modeling with undecimated fully convolutional neural networks”. In: *arXiv preprint arXiv:1508.00317* (2015).

[20] Chris Olah. *Understanding LSTM Networks – colah’s blog.* URL: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` (visited on 05/07/2017).

[21] Andrius Petrenas et al. “An echo state neural network for QRST cancellation during atrial fibrillation”. In: *IEEE Transactions on Biomedical Engineering* 59.10 (2012), pp. 2950–2957.

[22] Helmut Pürerfellner et al. “P-wave evidence as a method for improving algorithm to detect atrial fibrillation in insertable cardiac monitors”. In: *Heart Rhythm* 11.9 (2014), pp. 1575–1583.

[23] James A Reiffel et al. “Practice patterns among United States cardiologists for managing adults with atrial fibrillation (from the AFFECTS Registry)”. In: *The American journal of cardiology* 105.8 (2010), pp. 1122–1129.

[24] Juan Ródenas et al. “Wavelet Entropy Automatically Detects Episodes of Atrial Fibrillation from Single-Lead Electrocardiograms”. In: *Entropy* 17.9 (2015), pp. 6179–6199.

[25] Adam Santoro et al. “One-shot Learning with Memory-Augmented Neural Networks”. In: *arXiv:1605.06065 [cs]* (May 2016). arXiv: 1605.06065. URL: `http://arxiv.org/abs/1605.06065` (visited on 05/07/2017).

[26] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *arXiv:1409.1556 [cs]* (Sept. 2014). arXiv: 1409.1556. URL: `http://arxiv.org/abs/1409.1556` (visited on 05/09/2017).

[27] Andreas Veit, Michael Wilber, and Serge Belongie. “Residual Networks Behave Like Ensembles of Relatively Shallow Networks”. In: *arXiv:1605.06431 [cs]* (May 2016). arXiv: 1605.06431. URL: `http://arxiv.org/abs/1605.06431` (visited on 05/09/2017).

[28] Oriol Vinyals et al. “Matching Networks for One Shot Learning”. In: (June 2016). URL: `file:///home/csbotos/.mozilla/firefox/l9sv74v9.default/zotero/storage/47TNEP77/1606.html` (visited on 05/07/2017).

[29] Li Wan et al. “Regularization of neural networks using dropconnect”. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013, pp. 1058–1066. URL: `http://machinelearning.wustl.edu/mlpapers/papers/icml2013_wan13` (visited on 05/08/2017).

[30] Zhiguang Wang, Weizhong Yan, and Tim Oates. “Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline”. In: *arXiv preprint arXiv:1611.06455* (2016).

[31] Markus Waser and Heinrich Garn. “Removing cardiac interference from the electroencephalogram using a modified Pan-Tompkins algorithm and linear regression”. In: *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*. IEEE. 2013, pp. 2028–2031.

[32] Jason Yosinski et al. “Understanding neural networks through deep visualization”. In: *arXiv preprint arXiv:1506.06579* (2015).

[33] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 818–833.