Linear Approximation of Deep Neural Networks for Efficient Inference on Video Data

Bodo Rueckauer
Institute of Neuroinformatics
University of Zurich and ETH Zurich
Zurich, Switzerland
rbodo@ini.uzh.ch

Shih-Chii Liu
Institute of Neuroinformatics
University of Zurich and ETH Zurich
Zurich, Switzerland
shih@ini.uzh.ch

***Abstract*—Sequential data such as video are characterized by spatio-temporal correlations. As of yet, few deep learning algorithms exploit them to decrease the often massive cost during inference. This work leverages correlations in video data to linearize part of a deep neural network and thus reduce its size and computational cost. Drawing upon the simplicity of the typically used rectifier activation function, we replace the ReLU function by dynamically updating masks. The resulting layer stack is a simple chain of matrix multiplications and bias additions, that can be contracted into a single weight matrix and bias vector. Inference then reduces to an affine transformation of the input sequence with these contracted parameters. We show that the method is akin to approximating the neural network with a first-order Taylor expansion around a dynamically updating reference point. The proposed algorithm is evaluated on a denoising convolutional autoencoder.**

***Index Terms*—Deep neural networks, video, sequential data, linearization, compression**

I. Introduction

Sequential data in natural scenes typically changes little between samples, especially in video recordings of surveillance or highway driving. The high degree of correlation between consecutive samples implies that an almost linear model can be used for a finite sequence of frames. This property would benefit current deep neural networks (DNNs) used to process such video sequences. State-of-the-art DNNs for visual tasks consist of millions of neurons, and hold tens of millions of parameters. Computing the output of anyone of these networks for a single input frame requires billions of floating point multiply-accumulate operations (MACs). This substantial inference cost of neural networks on video is a major limiting factor for their use in mobile devices and always-on scenarios.

Based on the assumption of slowly changing input in video tasks, this paper proposes a novel method to linearize part of a DNN or an entire DNN around a dynamically updating reference point, similar to a first-order Taylor expansion of the network function. The resulting DNN is compressed in terms of number of neurons, parameters, and operations, allowing for efficient inference on sequential data.

This work has been supported by the Samsung Advanced Institute of Technology, the ETH Zurich and the University of Zurich.

Our network approximation method draws upon two common characteristics of DNNs: 1) The composite structure of neural networks where the mathematical function of a layer consists of a matrix multiplication with the output of the previous layer. 2) The simplicity of the frequently used Rectifying Linear Unit (ReLU) activation function. With the element-wise activation function replaced by a mask vector, as proposed here, we exploit the multiplicative nature of DNN layers to contract a stack of layers into a single layer. This layer, represented by a single weight matrix, approximates the function of the original layer stack around a reference point. The reference point, along with the contracted weight matrix, can be updated when temporal changes in the input sequence become large. These updates are expected to be sparse in slowly varying scenes. In the meantime, inference is done cheaply using the contracted weight matrix. We apply the proposed linearization and compression method to an autoencoder CNN for denoising temporal MNIST digits.

II. Related work

1) Network compression: Several groups have explored methods to make DNNs more efficient in processing sequential data by drawing upon redundancies in the input stream.

In the context of training recurrent neural networks (RNNs), the authors of Skip RNN [1] employ a control variable for each neuron that determines whether the state is updated or copied over from the previous time step. A regularization term encourages the model to use a reduced number of state updates during training. This update skipping can be seen as a zero-order approximation: The state is kept constant (copied over). In our case, replacing nonlinearities with masks still allows changes to be propagated through these masks, making it a first-order approximation.

Delta RNNs [2] capitalize on the stability of RNN activation patterns by transmitting neuron activations only when exceeding a threshold. Similarly, change-based CNNs [3] exploit spatio-temporal sparsity of pixel changes in video data to skip computations of neurons when their activation level changes by less than a certain threshold across frames. The resulting accuracy drop is proportional to the targeted efficiency gain and the difficulty of the dataset. In contrast, our method does not discard small activation changes via a local threshold; all changes are propagated through the mask that replaces the nonlinearity.

Other methods to compress deep neural networks for efficient inference include, low-precision models [4]–[6], pruning of redundant weights [7], and training convolutional architectures with fewer connections [8], [9].

2) Interpolation between key frames: An elegant framework for video recognition has been developed by [10]. It runs the DNN on sparse key frames and propagates their deep feature maps to other frames via a flow field. The optical flow is computed with a CNN that is trained end-to-end with the video recognition model. The method is similar to ours in the use of key frames, but differs in how inference is done between key frames (propagating hidden layer states along flow vectors, versus linearizing the network function and contracting a stack of layers). Also, while the authors of [10] update the key frames at regular intervals, we also test an input-driven update predictor. Lastly, their method makes explicit use of temporal correlation in video via optical flow whereas we implicitly draw on this assumption to motivate a linear approximation.

3) Taylor expansions in DNNs: We show in Sec. III that our network contraction method can be seen as a first order Taylor expansion, where the derivatives are taken with respect to the network input. Taylor expansions have been applied to neural networks to explain nonlinear classification decisions [11], to generate a class saliency map of a specific input [12], and to analyze the learning convergence under various optimizers [13].

III. Methods

A. Activation masking and network contraction

In this work, we consider only feed-forward deep neural networks. Each neuron in a network layer receives as input, the linear combination of the outputs of the preceding layer and applies a nonlinear activation function on this weighted sum of inputs. For instance, an n-layer fully-connected ANN has the following computational structure:

$$F(\mathbf{x}) = W^n f\big(W^{n-1} \dots f\left(W^1\mathbf{x} + \mathbf{b}^1\right) + \dots + \mathbf{b}^{n-1}\big) + \mathbf{b}^n, \tag{1}$$

where F is the network output, $\mathbf{x}$ is the network input, pairs $(W^k, \mathbf{b}^k)$ represent the weights and biases of layer $k \in [1, n]$, and f is the nonlinear activation function of a neuron. Typically, the network output is also passed through a final nonlinearity f_{out}, e.g. a softmax in a classification task. To simplify notation in the following equations, we do not explicitly write this output nonlinearity, but imply that it is applied as usual after a forward-pass through the network. A common nonlinearity for hidden layers is the ReLU, and its variant, the leaky ReLU:

$$f(z_i^k) = \begin{cases} z_i^k & \text{if } z_i^k > 0 \\ \alpha z_i^k & \text{otherwise.} \end{cases} \tag{2}$$

The variable z_i^k stands for the pre-activation of neuron i in layer k, i.e. the summed output before applying the nonlinearity. The leak parameter, α, defines the slope of the left branch and typically has a small value (e.g. $\alpha = 0.1$). In the case of the standard ReLU, $\alpha = 0$.

From (2), it is apparent that the element-wise application of the (leaky) ReLU f in (1) is equivalent to the element-wise multiplication with a mask $\mathbf{m}^k$:

$$F(\mathbf{x}) = W^n \mathbf{m}^{n-1} \odot \big(W^{n-1} \dots \mathbf{m}^1 \odot \left(W^1\mathbf{x} + \mathbf{b}^1\right) + \dots + \mathbf{b}^{n-1}\big) + \mathbf{b}^n \tag{3}$$

where $\odot$ denotes the Hadamard product. In this work, the Hadamard product takes precedence over the matrix-multiplications, i.e. the mask $\mathbf{m}^k$ is applied to the pre-activation of layer k before multiplication with the weights W^{k+1} of layer $k+1$. The mask $\mathbf{m}^k$ of layer k is a vector whose length is equal to the number of neurons in layer k. Its entries are defined by[1]

$$m_i^k = \begin{cases} 1 & \text{if } z_i^k > 0 \\ \alpha & \text{else.} \end{cases} \tag{4}$$

Because matrix multiplications are associative, we can contract any number of matrix-, mask-, and vector-products in (4) into a single weight matrix Q and bias vector $\mathbf{q}$ (see also Fig. 1a):

$$F(\mathbf{x}) = Q\big|_{\mathbf{x}}\mathbf{x} + \mathbf{q}\big|_{\mathbf{x}}, \text{ where} \tag{5}$$

$$\begin{aligned} Q\big|_{\mathbf{x}} &= W^n \mathrm{diag}(\mathbf{m}^{n-1}) \odot W^{n-1} \dots \mathrm{diag}(\mathbf{m}^1) \odot W^1, \\ \mathbf{q}\big|_{\mathbf{x}} &= W^n \mathbf{m}^{n-1} \odot \big(W^{n-1} \dots \mathbf{m}^1 \odot \mathbf{b}^1 + \dots + \mathbf{b}^{n-1}\big) + \mathbf{b}^n. \end{aligned} \tag{6}$$

The key idea now is to use a sample $\mathbf{x}_*$ to compute the $(Q, \mathbf{q})\big|_{\mathbf{x}_*}$ parameters once, and then to perform inference on the next few samples in the sequence by using the much simplified affine transformation (5).

This process is related to the linearization of a complicated function (in our case the network $F(\mathbf{x})$) via Taylor expansion around a reference point $\mathbf{x}_*$:

$$T_F(\mathbf{x}) = F(\mathbf{x}_*) + J_F\big|_{\mathbf{x}_*}(\mathbf{x} - \mathbf{x}_*) + \dots, \tag{7}$$

with $J_F\big|_{\mathbf{x}_*} = \frac{\partial F}{\partial \mathbf{x}}\big|_{\mathbf{x}_*}$ the Jacobian of F evaluated at $\mathbf{x}_*$. We determine the conditions under which our contracted network $C_F(\mathrm{x})$, defined by

$$C_F(\mathrm{x}) = Q\big|_{\mathbf{x}_*}\mathbf{x} + \mathbf{q}\big|_{\mathbf{x}_*}, \tag{8}$$

resembles a Taylor approximation around the reference point $\mathbf{x}_*$. By construction (c.f. (1)), the contracted network exactly equals the original network at the reference point:

[1]Because the mask depends on the current input sample, it could be written as $\mathbf{m}^k\big|_{\mathbf{x}}$. We omit this subscript here for clarity and introduce it later.

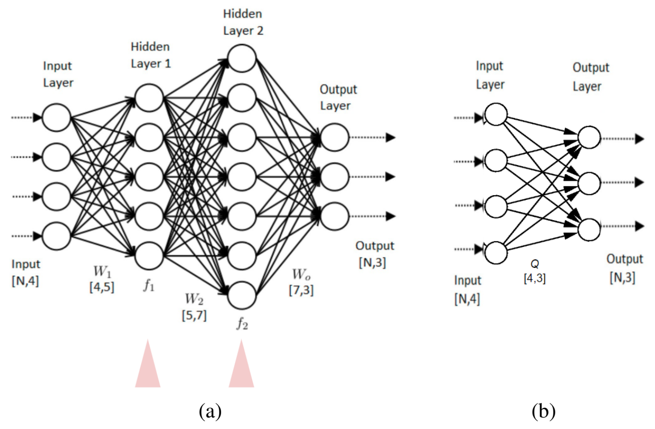

Fig. 1. Illustration of network masking and contraction. (a) Activation functions are first replaced by dynamically updating masks (triangles, cf. Eq. 3). (b) Hidden layer weight and mask matrices are then contracted (Eq. 5).

$$C_F(\mathrm{x}_*) = Q\big|_{\mathbf{x}_*}\mathbf{x}_* + \mathbf{q}\big|_{\mathbf{x}_*} = F(\mathbf{x}_*). \tag{9}$$

By using identity (9) to replace the zero-order term in (7), and after some rearranging, we obtain:

$$T_F(\mathbf{x}) = \mathbf{q}\big|_{\mathbf{x}_*} + \Big(Q\big|_{\mathbf{x}_*} - J_F\big|_{\mathbf{x}_*}\Big)\,\mathbf{x}_* + J_F\big|_{\mathbf{x}_*}\mathbf{x} + \dots . \tag{10}$$

Comparing coefficients of (8) and (10), we can see that the contracted network is identical with a first-order Taylor expansion if Q is equal to the Jacobian:

$$C_F(\mathbf{x}) = T_F^{(1)}(\mathbf{x}) \quad \text{if} \quad Q\big|_{\mathbf{x}_*} = J_F\big|_{\mathbf{x}_*}. \tag{11}$$

In Sec. IV, we will show that this relation holds through our experiments with a denoising autoencoder.

Having established a relationship between the contracted network and the first-order Taylor approximation of the original network, we now consider when to update the Q matrix at a new reference point $\mathbf{x}_*$. At $\mathbf{x}_*$, the simplified affine transformation (5) is exactly equivalent to a standard forward-pass through the original multi-layer network. As the input samples begin to deviate from the reference point, the masks used for computing the contracted $(Q, \mathbf{q})\big|_{\mathbf{x}_*}$ parameters remain accurate only under the condition that the neurons do not change the sign of their activity $\mathbf{z}$ (c.f. (4)). For instance, if a neuron i in layer k changes from being positively activated at sample $\mathbf{x}_* = \mathbf{x}_t$ to being negatively activated at sample $\mathbf{x}_{t+1}$, then the corresponding entry in the mask vector m_i^k should be α instead of 1, and the $(Q, \mathbf{q})\big|_{\mathbf{x}_t}$ parameters introduce an error. Note that such errors occur exclusively at sign changes: Otherwise, the $(Q, \mathbf{q})\big|_{\mathbf{x}_t}$ parameters accurately encode the network function even under arbitrarily large changes of z_i^k.

To keep the network operation accurate even when activation signs are changing over the course of presenting an input sequence, it is necessary to recompute the $(Q, \mathbf{q})$ parameters at appropriate intervals, using updated binary masks that represent the state of network activations at the new reference point. Thus, we face an accuracy-efficiency trade-off: The less often the update of the $(Q, \mathbf{q})$ parameters, the lower is the run-time cost of inference, but the higher is the risk of inaccurate activation masks.

The most straight-forward baseline criterion for mask updates is the use of a regular update interval. The hyperparameter in this case is the number of frames, n, before the mask is updated. The advantages of using this criterion are that the network output is easy to interpret and n can be related to the expected savings in computations. Further, the criterion offers a safety guarantee by limiting the longest period without updates to n frames.

A more flexible criterion that accounts for direct changes in input is by computing the pixel-wise square difference on input images, i.e., the mean-square error (MSE) between the current frame and the frame at the time of the last update. If the MSE value surpasses a given threshold, a mask update is triggered. These two update criteria are compared in Sec. IV and Fig. 4.

IV. Results

The network contraction method is implemented in Python using Keras with the TensorFlow backend. We apply the network contraction on a denoising task using a convolutional autoencoder. The input sequence is a version of the MNIST handwritten-digit dataset [14] where the images are sorted by similarity (according to the pixel-wise mean-square-error). The encoder section of the denoising autoencoder (DAE) consists of two convolution layers followed by a fully-connected layer, mapping the 28×28 gray-scale input images into a 16-dimensional latent space. The decoder section consists of a fully-connected layer and three transpose-convolution layers, which map the 16-dimensional encoded representation back into image space. In the denoising task, the input to the DAE consists of the temporal MNIST digits corrupted by adding normally distributed noise with standard deviation 0.5, and the outputs are the denoised digits. The uncorrupted MNIST samples serve as targets when training the DAE, and as ground-truth at test time.

In the DAE experiment, we contract the decoder and leave the encoder unchanged. The resulting architecture then consists of two convolution layers and two fully-connected layers, the last being the contracted decoder.

In Fig. 2, we count the number of activation sign changes in the DAE between successive frames in the temporal MNIST dataset. Each curve represents a layer, color-coded from dark blue (lowest layer) to dark red (highest layer). Depicted are the two convolution layers of the encoder (dark and light blue), and the first two convolution layers of the decoder (light and dark red). The other three layers of this architecture are not shown because they do not have a ReLU activation viable for masking. We first note that the number of activation sign changes in the two transposed convolution layers of the decoder are correlated. This is advantageous for network contraction because it implies that the need to update masks is synchronized across layers. Secondly, we observe a trend as in

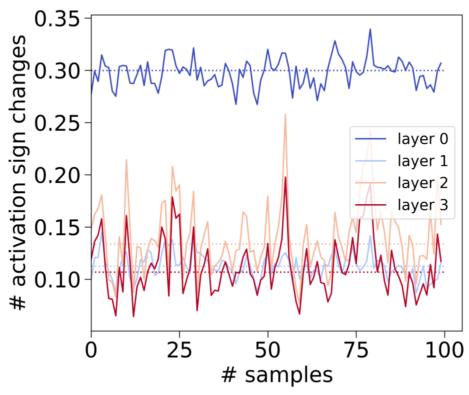

Fig. 2. Number of activation sign changes in the Denoising Auto-Encoder architecture for the first 100 samples of the temporal MNIST dataset. The dotted lines denote the median.

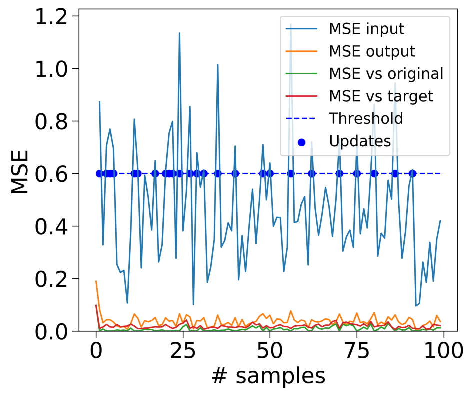

Fig. 3. Mask update indicators and accuracy metrics in the Denoising Auto-Encoder architecture for the first 100 samples of the temporal MNIST dataset.

[15], where *contiguity* (the length of sequences of activations with the same sign) increases with network depth. That is, the lowest convolution layer in the encoder changes signs much more often than higher layers, and the second transposed convolution layer in the decoder has consistently fewer sign changes than the first. These observations support our choice to contract the decoder, but not the encoder.

In Fig. 3, we measure the mask update indicator *MSE input* described in Sec. III. In this case, *MSE input* does not refer to the noise-corrupted MNIST digits fed to the encoder, but the latent representations fed to the decoder. The plot also shows the MSE between output frames (*MSE output*), the MSE between the output of the contracted model and the original model (*MSE vs original*) and the MSE between the contracted model and ground-truth (*MSE vs target*).

To examine if the *MSE input* metric is a good predictor of when the mask updates are needed, we correlate the *MSE input* curve in Fig. 3 with the number of activation sign changes in Fig. 2 across the length of the sequence. The obtained Pearson correlation coefficients of 0.8 and 0.78 in the two transposed convolution layers of the decoder indicate a good correlation between the input MSE and the actual need to update masks, as measured by the number of activation sign changes.

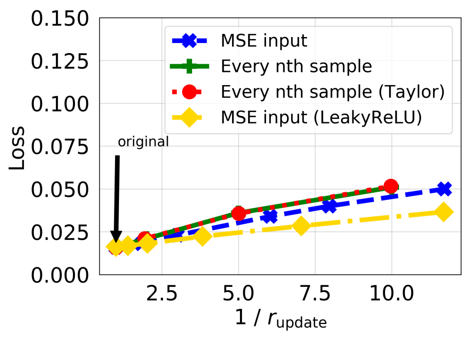

Fig. 4. Loss (reconstruction MSE) plotted against the inverse mask update rate for the DAE task on temporal MNIST.

Figure 4 illustrates the accuracy-efficiency trade-off for the contracted DAE on temporal MNIST. The mask update rate is controlled either by the hyperparameter n in case of updates at every n-th frame, or by varying the threshold on the input MSE. The fewer updates of the masks, the more savings in computations (horizontal axis), and the higher the loss (vertical axis). With regular mask update intervals (solid line), the loss is consistently higher than the loss using the input MSE as the update indicator (dashed line), highlighting the benefit of a dynamic update criterion. Also shown is the performance of a network trained with leaky ReLUs (diamond markers). This model performs better because the leak makes the contracted network more robust against outdated masks. The fourth curve (dotted line) results from the first-order Taylor expansion of the original network, and performs the same as the contracted model. This experimental result supports the theoretical relation of our method with the Taylor approximations derived in Sec. III-A.

Contracting the decoder reduces its number of parameters by more than $8\times$ (from 109 k to 13 k), the number of neurons by more than $57\times$ (from 45 k to 784), and the number of operations by $1692\times$ (from 44 M to 26 k). Despite this significant compression, the contracted DAE maintains a good denoising performance, as seen in the output video[2]. When mask updates are triggered by *MSE input* on average every third frame, the reconstruction MSE between the output of the contracted model and the ground-truth is 0.023 (0.016 for the original model).

Autoencoders are well-suited for network contraction because the latent representation of the DAE architecture is low-dimensional, resulting in a negligible inference cost of the contracted network. Another possible benefit of this architecture is that much of the spatial dynamics of the input are absorbed in the latent representation by the encoder, reducing the need for updating masks often in the decoder.

V. Conclusion

Deep networks are used in numerous applications that require analysis of video data, e.g. smart homes, personal assistance, surveillance etc. Running such networks continuously

[2]https://youtu.be/bBFTkyckQe8

(as in surveillance) can be costly in terms of both hardware requirements and energy consumption. This substantial inference cost of neural networks on video is a major limiting factor for their use in mobile devices and always-on scenarios. Our method dynamically contracts a stack of layers into a single layer, thereby reducing the complexity and computational cost of the network in situations where the scene hardly changes between two frames. Mask updates can be performed at regular intervals and can also be triggered dynamically following an input-dependent metric.

Our results on a convolutional autoencoder for denoising temporal MNIST show that the contracted model is equivalent to a first-order Taylor expansion with updating reference point. Approximation of the network via masking and contraction, results in a condensed network that requires fewer computations during inference. The proposed method may thus be beneficial for reducing the run-time cost of deep neural networks on sequential data like surveillance and highway-driving.

Future work includes extending the method to other architectures and datasets, exploring different mask update predictors, and quantifying the computational cost of the network contraction.

References

[1] V. Campos, B. Jou, X. Giro-i Nieto, J. Torres, and S.-F. Chang, "Skip RNN: Learning to Skip State Updates in Recurrent Neural Networks," in *International Conference on Learning Representations*, 2018, pp. 1–17. [Online]. Available: http://arxiv.org/abs/1708.06834

[2] D. Neil, J. H. Lee, T. Delbruck, and S.-C. Liu, "Delta Networks for Optimized Recurrent Network Computation," in *PMLR*, 2017. [Online]. Available: http://arxiv.org/abs/1612.05571

[3] L. Cavigelli, P. Degen, and L. Benini, "CBinfer: Change-Based Inference for Convolutional Neural Networks on Video Data," *arXiv*, 2017. [Online]. Available: http://arxiv.org/abs/1704.04313

[4] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "WRPN: Wide Reduced-Precision Networks," *arXiv*, pp. 1–11, 2017. [Online]. Available: http://arxiv.org/abs/1709.01134

[5] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional Neural Networks using Logarithmic Data Representation," *arXiv*, 2016. [Online]. Available: http://arxiv.org/abs/1603.01025

[6] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations," *arXiv:1602.02830*, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[7] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in *ICLR*, 2015, pp. 1–13. [Online]. Available: http://arxiv.org/abs/1510.00149

[8] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size," *arXiv*, pp. 1–5, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[9] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861*, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[10] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 4141–4150, 2017.

[11] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, 2017.

[12] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[13] D. Balduzzi, B. McWilliams, and T. Butler-Yeoman, "Neural Taylor Approximations: Convergence and Exploration in Rectifier Networks," in *ICML*, 2016. [Online]. Available: http://arxiv.org/abs/1611.02345

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86. IEEE, 1998, pp. 2278–2323.

[15] D. Balduzzi, M. Frean, L. Leary, J. Lewis, K. W.-D. Ma, and B. McWilliams, "The Shattered Gradients Problem: If resnets are the answer, then what is the question?" in *ICML*, 2017. [Online]. Available: http://arxiv.org/abs/1702.08591