STABLE AND SYMMETRIC FILTER CONVOLUTIONAL NEURAL NETWORK

Raymond Yeh, Mark Hasegawa-Johnson, Minh N. Do

University of Illinois at Urbana Champaign, Champaign, IL, USA
Department of Electrical and Computer Engineering
{yeh17, jhasegaw, minhdo}@illinois.edu

ABSTRACT

First we present a proof that convolutional neural networks (CNN) with max-norm regularization, max-pooling, and Relu non-linearity are stable to additive noise. Second, we explore the use of symmetric and antisymmetric filters in a baseline CNN model on digit classification, which enjoys the stability to additive noise. Experimental results indicate that the symmetric CNN outperforms the baseline model for nearly all training sizes and matches the state-of-the-art deep-net in the cases of limited training examples.

Index Terms— Convolutional Neural Network, Symmetric, Deep Learning

1. INTRODUCTION

Deep convolutional neural networks have demonstrated impressive performance on image classification [1]. There are many studies attempting to understand the effectiveness of these networks both empirically by visualizing the activations, and theoretically through analyzing the network properties [2, 3, 4]. In particular, Bruna and Mallat have demonstrated the effectiveness of translation invariance and stability to additive noise on a scattering net, a specified type of convolutional network [5, 6]. In this paper, we use the framework proposed by Mallat to prove that CNNs with max-norm regularization, max-pooling, and Relu non-linearity operations are stable to additive noise. The effectiveness of max-norm regularization has been shown in context of collaborative filtering [7] and deep learning [8]. This proof provides an explanation for the max-norm regularization, max-pooling, Relu non-linearity, and convolutional neural networks are a reasonable inductive bias for image classification.

Scattering net with hand-crafted wavelets as the weights for its network, which is highly structured, has achieved state-of-the-art results on digit classification [5]. This suggests that a fully trainable CNN might have an over complex hypothesis space for this task. As a compromise between fully trained and fully untrained filters, we propose to restrict the hypothesis space to CNNs with only symmetric or antisymmetric weights architecture. With this setup, we have half the number of weights to train in the convolution layer, ensure linear phase filter response, and gain the potential to accelerate the training and decoding process as the convolution operator with symmetric filters could be computed more efficiently.

2. CNN'S STABILITY TO ADDITIVE NOISE

For a transformation, Φ, to be stable to additive noise $x'(u) = x(u) + \epsilon(u)$, it needs a Lipschitz continuity condition as defined in [5],

$$\|\Phi x - \Phi x'\|_2 \leq C \cdot \|x - x'\|_2 \tag{1}$$

for a constant $C > 0$, and for all x and x'. Φx denotes the transformed feature.

Next, a standard CNN has the following forward operations, 1) convolution with max-norm regularization, 2) element-wise Relu non-linearity, and 3) max-pooling. We will prove that the sequence of these operations can satisfy the Lipschitz continuity condition.

2.1. Stability of convolution with max-norm regularization

Denote the output of a convolution as $w \star x$. With l_1 max-norm regularization, the weights are renormalized to constant norm, κ. This means $\|w\|_1 \leq \kappa$. By Young's inequality for convolutions [9],

$$\|w \star x\|_2 \leq \|w\|_1 \cdot \|x\|_2 \leq \kappa \cdot \|x\|_2 \tag{2}$$

Then by linearity of convolution

$$\|w \star x - w \star x'\|_2 \leq \kappa \cdot \|x - x'\|_2 \tag{3}$$

If x is multi-channel, the convolutional layer sums the convolution outputs of each channel. Then, by triangle inequality, Lipschitz continuity condition holds with $C = \kappa \cdot$ (number of channels).

2.2. Stability of Relu operation

Relu is an element-wise operation defined as

$$Relu(x_i) = max(0, x_i) \tag{4}$$

where x_i denotes an element in the input signal x. Next, it can be verified that $|Relu(x_i) - Relu(x'_i)| \leq |x_i - x'_i|$, by considering the four cases with x_i and x'_i each positive or negative. Therefore, $\|Relu(x) - Relu(x')\|_2 \leq \|x - x'\|_2$.

2.3. Stability of max-pooling

Max-pooling operation divides the input signal into a set of overlapping or non-overlapping windows, and for each window outputs the maximum value.

First consider the windows to be non-overlapping, then we only need to show that the max operation for each window, following a Relu operation, is Lipschitz continuous. Recall that, max-pooling operation follows the Relu operation, and therefore $x > 0$ and $x' > 0$, where x and x' denote the signals in each window. Denote $i^* = \arg\max_i x_i$ and $j^* = \arg\max_j x'_j$. We claim that

$$|\max_i x_i - \max_j x'_j| \leq \max(|x_{i^*} - x'_{i^*}|, |x_{j^*} - x'_{j^*}|) \quad (5)$$

Then,

$$\begin{aligned} |\max_i x_i - \max_j x'_j| &\leq \max_i |x_i - x'_i| \\ &= \|x - x'\|_\infty \leq \|x - x'\|_2 \end{aligned} \quad (6)$$

If the inequality is true, then the max-pooling operator satisfies Lipschitz continuous condition. The inequality can be proved by considering the following two cases.

1. $x_{i^*} > x'_{j^*} \rightarrow |\max_i x_i - \max_j x'_j| = x_{i^*} - x'_{j^*} \leq x_{i^*} - x'_{i^*}$
   As x and x' are all greater than 0 and x'_{j^*} is the largest in x'.
2. $x'_{j^*} > x_{i^*} \rightarrow |\max_i x_i - \max_j x'_j| = x'_{j^*} - x_{i^*} \leq x'_{j^*} - x_{j^*}$

Therefore, max-pooling operation with non-overlapping windows satisfies the Lipschitz continuity condition with $C = 1$

Next, consider overlapping windows, with k overlaps; where k is less than the window size. As each of the overlapping term's contribution to the norm is less than or equal to $\|x - x'\|_2$, we can show that

$$\|maxPool_k(x) - maxPool_k(x')\|_2 \leq (k+1) \cdot \|x - x'\|_2 \quad (7)$$

, where k is the number of overlapping elements in the pooling window.

3. SYMMETRIC FILTER CNNS

The symmetric filter CNN is motivated by recent results showing that a Scattering Net, with weights set equal to wavelet coefficients and untrained, was able to reach state of the art performance in handwritten digit recognition [5]. As wavelets have symmetric or antisymmetric structure, we speculate that the hypothesis space of the CNN model can be restricted to only symmetric and antisymmetric convolution layers. Let W denote a weight(filter) coefficients centered at $(0, 0)$. By antisymmetric we mean $W(i, j) = -W(-i, -j)$ and by symmetric we mean $W(i, j) = W(-i, -j)$.

Symmetric and antisymmetric filters with odd height and width have generalized linear phase. This ensures that no phase distortion occurs at the convolutional layer, hence the structure of the signal is maintained; this is a very common practice in filter design [10]. Furthermore, when enforcing this symmetric constraint, the number of parameters to train is reduced and the potential to accelerate training and decoding by using a symmetric convolution operator is gained, as convolution on symmetric filters requires half as many multiplications as convolution with arbitrary filters.

3.1. Symmetry in image recognition tasks

At first glance a constraint of symmetric and antisymmetric filters with respect to the origin seems like a very strong condition. However, the overall model can represent approximately the same set of functions as a model that has symmetric or antisymmetric weights with respect to a certain point, not necessarily the origin; many of the learned CNN filters published as examples in image recognition papers have approximately this property, as do the many of the filters learned in our own baseline experiments. The reasoning is as follows, denote the translation operator $T_{\vec{c}}$, such that $T_{\vec{c}}(W(\vec{x})) = W(\vec{x} - \vec{c})$. Then, for a filter symmetric to some point $\vec{c}$, we can translate the filter to be centered at the origin. From the translation invariance property of convolution, $T_{\vec{c}}(W) \star g = T_{\vec{c}}(W \star x)$, i.e., the output from the convolution layer is translated. Therefore, we see that centering the filter will result in a translated output and no loss of information.

Relu is an element-wise operator, and thus the output continues to be a translated version. For max-pooling of window size $N \times N$, if $\vec{c}$ happens to be a multiple of N, then the output from the centered model will again be the translated version of the non-centered model. On the other hand, $\vec{c}$ is less than N, then as long as the max element does not move out of the max-pooling window, the output will be equivalent; hence the output is approximately the same as the one from the non-centered model. Lastly, the fully connected layers are not affected by reordering of the inputs, as reordering the weights in the same manner will give equivalent output.

3.2. Model architecture

The model architecture we used is shown in Fig. 1. In order to have a direct observation on the effect of the proposed symmetric filter CNN model, we have chosen a simple network architecture modified from LeNet [11], using Relu as

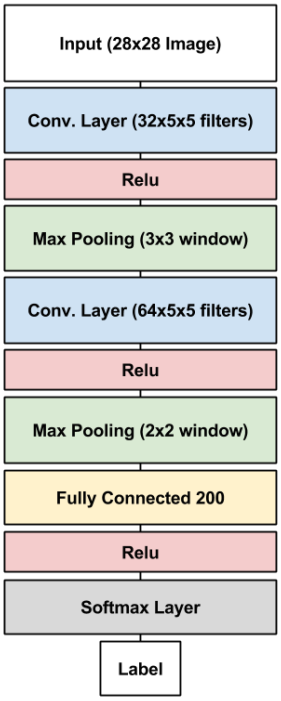

Fig. 1: CNN architecture used in experiments

the non-linearity operation, and max-pooling with 2x2 non-overlapping windows. We used this model as the baseline benchmark. Experiment results for the comparison are shown in Table 1.

The symmetric filter CNN model follows the same architecture as the baseline described, except for the change that at each convolutional layer, half of the filters are enforced to be symmetric and the others to be antisymmetric.

The model weights were randomly initialized. No pre-training or dropout were used.

4. EXPERIMENTS

4.1. MNIST dataset

The MNIST database of hand-written digits contains 60,000 training samples and 10,000 test samples [12]. We evaluated on different training sizes and report the results in Table 1. For each training size, we randomly sampled from the training set with a constraint that each digit occurs with the same number of times [5], which is to avoid very skewed distribution possibly resulting from random sampling. Also, no distortion of any kind was used to enhance the training data; preprocessing done to the data was normalization.

4.2. Learning procedure

We trained our networks using stochastic gradient descent with momentum. We used 10,000 random examples from the training data as a holdout set for tuning hyperparameters; this includes learning rate, regularization parameters (max-norm and l_2 regularization), momentum and batch size. The tuning procedure follows the suggested techniques mentioned in [13]. The identical tuning procedure is performed on the baseline model and the symmetric model, to control for the effect of tuning on the performance of the models.

4.2.1. Gradient for symmetric convolutional layer

In this section, we derive the gradient formula for the symmetric/antisymmetric convolutional layer, using back propagation notation of [14]. First denote the following terms,

1. $a_j^{(l)} = j^{th}$ channel of the activation map at l^{th} layer.
2. $W_{ij}^{(l)} = i^{th}$ channel of the j^{th} filter at l^{th} layer.
3. $f(\cdot) =$ an element-wise non-linear operator.
4. $\star =$ convolution.
5. $J =$ overall loss function
6. $\delta_j^{(l)}(u,v) = \frac{\partial J}{\partial z_j^{(l)}(u,v)} =$ backprop error.

The forward convolutional operation without symmetric constraint can be defined as

$$z_j^{(l)} = (\sum_i a_i^{(l-1)} \star W_{ij}^{(l-1)}) \tag{8}$$

$$a_j^{(l)} = f(z_j^{(l)}) \tag{9}$$

Then the gradient of J with respect to the filter weight is

$$\frac{\partial J}{\partial W_{ij}^{(l-1)}(u,v)} = \sum_{u'}\sum_{v'} \frac{\partial J}{\partial z_j^{(l)}(u',v')} \cdot \frac{\partial z_j^{(l)}(u',v')}{\partial W_{ij}^{(l-1)}(u,v)} \tag{10}$$

From equation 9, we can see that

$$\frac{\partial z_j^{(l)}(u',v')}{\partial W_{ij}^{(l-1)}(u,v)} = a_i^{(l-1)}(u'-u, v'-v) \tag{11}$$

as the gradient is non-zero when $\hat{u} = u$, and $\hat{v} = v$. Lastly, for the simplicity of indexing, let $\widetilde{W}_{ij}^{l-1}(u,v)$ be the weights of the symmetric convolution layer, constrained so $\tilde{W}_{ij}^{(l-1)}(-u,-v) = \tilde{W}_{ij}^{(l-1)}(u,v)$.

The gradient with respect to the symmetric filter weight can be written in terms of the gradient of the general convolution layer as follows,

$$\frac{\partial J}{\partial \widetilde{W}_{ij}^{(l-1)}(u,v)} = \sum_{u'}\sum_{v'} \frac{\partial J}{\partial z_j^{(l)}(u',v')} \cdot \frac{\partial z_j^{(l)}(u',v')}{\partial W_{ij}^{(l-1)}(u,v)} + \sum_{u'}\sum_{v'} \frac{\partial J}{\partial z_j^{(l)}(u',v')} \cdot \frac{\partial z_j^{(l)}(u',v')}{\partial W_{ij}^{(l-1)}(-u,-v)} \tag{12}$$

Training Size	Sym-Conv. Net	Base-line. Conv. Net	State-of-art Conv. Net
300	**9.95**	*10.30*	10.63
1,000	**4.31**	*4.40*	4.48
2,000	3.25	*3.20*	**3.05**
5,000	*2.15*	2.21	**1.98**
10,000	1.45	*1.30*	**0.84**
20,000	*1.01*	1.06	**0.70**
40,000	*0.82*	0.85	**0.64**
60,000	*0.70*	0.74	**0.62**

Table 1: Percentage of Errors on MNIST Test Set vs. Training Size

The gradient for the antisymmetric convolution layer can be derived similarly.

4.3. Results and Discussions

Table 1 reports the results from the symmetric convolutional network, the baseline network without symmetric filters, and the state-of-the-art convolutional network model (5-layers) with no pre-training, no image distortion, and no other improvement techniques [15], which is a reasonable comparison to our model.

The symmetric convolutional network outperforms the baseline model for nearly all training sizes; the difference in error rates between the symmetric and the baseline models decreases as training set size increases.

For small training size (e.g. 300 and 1,000), both the symmetric and baseline models outperform the state-of-the-art deep-net, when randomly initialized supervised in [15]. These results support the intuition that more complex models are more prone to overfitting, and simpler models perform better with limited training data.

Next, Fig. 2 presents the weights visualization of the first convolutional layer from the trained symmetric CNN on size 20,000, where the first four rows are for symmetric filters, and the bottom four rows are for antisymmetric ones. These weights are matched with our signal processing intuition. Denote (i, j) as the i^{th} row j^{th} column in Fig. 2. Consider $(3, 4)$, the middle 3×3 pixels resembles exactly a high pass filter. Overall, the weights are very interpretable, i.e. they are all roughly edge detections in a particular direction, which is very reasonable as the edges of a digit are likely the most discriminant classification features. Furthermore, we compared the weights learned from the symmetric CNN and baseline CNN, we observed that some of the weights, Fig. 3, are identical, but symmetric CNN has the weights centered at the origin, (e.g $(5, 1)$, $(5, 2)$), as suggested by the analysis in Section 3.1.

Lastly, we have also examined models with only symmetrical filters and with only antisymmetric filters. Overall, the convolutional network with half antisymmetric filters and half symmetric filters outperforms the models with only antisymmetric filters or with only symmetric filters. Furthermore, the network with only antisymmetric filters outperforms the one with only symmetric filters. These findings lead to the conclusion that antisymmetric filters are important for correctly identifying the digits, but antisymmetric filters are not sufficient without the complementary information provided by symmetric filters.

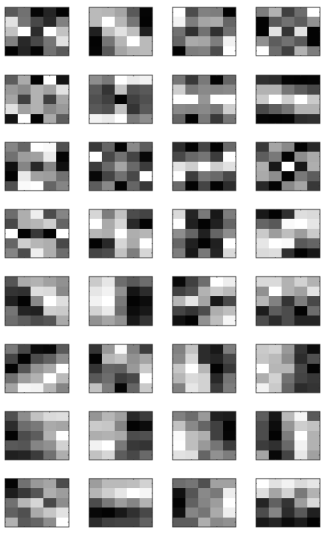

Fig. 2: First convolutional layer weights visualization from symmetric CNN

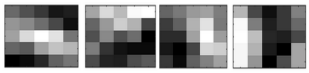

Fig. 3: Selected first convolutional layer weights visualization from baseline CNN

5. CONCLUSIONS

We present a proof that CNNs with max-norm regularization, Relu non-linearity, and max-pooling are stable to additive noise. We investigate the use of symmetric and antisymmetric filters in CNN model on the MNIST dataset. State-of-the-art results were achieved for handwritten digit classification in the cases of very small training sizes. We also show that the network with symmetric and antisymmetric filters is generally better than the baseline benchmark model. Lastly, we analyzed the model weights and verified our understanding that the set of functions that the symmetric models have learned are empirically similar to those of the baseline model.

6. REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] Matthew D Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014*, pp. 818–833. Springer, 2014.

[3] Ian Goodfellow, Honglak Lee, Quoc V Le, Andrew Saxe, and Andrew Y Ng, "Measuring invariances in deep networks," in *Advances in neural information processing systems*, 2009, pp. 646–654.

[4] Ian J Goodfellow and Oriol Vinyals, "Qualitatively characterizing neural network optimization problems," *arXiv preprint arXiv:1412.6544*, 2014.

[5] Joan Bruna and Stéphane Mallat, "Invariant scattering convolution networks," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1872–1886, 2013.

[6] Stéphane Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.

[7] Nathan Srebro and Adi Shraibman, "Rank, trace-norm and max-norm," in *Learning Theory*, pp. 545–560. Springer, 2005.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[9] William Beckner, "Inequalities in Fourier analysis," *Annals of Mathematics*, pp. 159–182, 1975.

[10] Jelena Kovacevic, Vivek Goyal, and Martin Vetterli, ," in *Fourier and Wavelet Signal Processing*. Cambridge University Press, 2014.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] Yann LeCun, Corinna Cortes, and Christopher JC Burges, "The MNIST database of handwritten digits," 1998.

[13] Yoshua Bengio, "Practical recommendations for gradient - based training of deep architectures," in *Neural Networks: Tricks of the Trade*, pp. 437–478. Springer, 2012.

[14] Christopher M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Inc., New York, NY, USA, 1995.

[15] Marc Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," pp. 1–8, 2007.