ADDING NON-LINEAR CONTEXT TO DEEP NETWORKS

Michele Covell, David Marwood, Shumeet Baluja

Google Research, 1600 Amphitheatre Parkway, CA 94043

ABSTRACT

Enormous success has been achieved with deep neural networks consisting of standard linear-convolutions followed by simple non-linear mapping functions. In this paper, we add easily-computed non-linear local and global statistics to deep architectures, augmenting the information available at each layer. This additional information is then used in an identical manner to current processing. The summary statistics, which can be as simple as calculating within-channel variance, introduces little run-time computational overhead and can be instantiated with few extra parameters. All standard training procedures can be used without modification for training these augmented networks. We show, through extensive testing with ResNet on ImageNet, performance improvements across a wide range of network sizes. Additionally, we provide a detailed study of where within the deep networks these statistics are most effective.

Index Terms— Deep neural networks, Non-linear layers, Image classification, ResNet

1. INTRODUCTION

Modern deep neural networks have convincingly shown the power of linear-layer computations followed by non-linearities. The convolutions and global average pooling common in vision tasks [1, 2] are limited to linear transformations. In this paper, we provide a simple method to augment existing convolutional and global-average-pooling layers with non-linear information inside the layer. We experiment with simple non-linear summary statistics on ResNet [3, 4]: standard deviation, variance, and L_1 distance. Despite their simplicity, they both improve performance in terms of classification and add little in terms of extra parameters or computation.

We use these three summary statistics in two novel layers that can be trivially added to the existing layers of any deep network:

- **NLConv**, a non-linear contextual layer, similar to a convolutional layer, for a local measure of spatial variation.
- **global NLPool**, similar to **NLConv** but giving a global measure of spatial variation.

In the next section, we briefly review related work and provide the necessary details of ResNet, the architecture on which we ran our experiments. Section 3 introduces our two layers. Section 4 presents detailed results of adding these non-linear computations to various parts of the ResNet architecture. Ablation studies are also shown. Section 5 provides a summary and directions for future work.

2. RELATED WORK

Non-linearities, like ReLU and tanh, are the main sources of expressive power for deep networks [5], but most of them are simple element-wise functions that treat each spatial sample separately. The most notable exceptions to this are network elements that feed spatial context into soft switches, such as LSTMs [6], transformers [7] and squeeze-excite blocks [8]. The softmax used in transformers, after the dot-product of the query and key vectors and before the weighted averaging of the value vector, change what would otherwise be a quadratic scaling of the input into a soft switch. Similarly, over much of their operating ranges the sigmoids in the squeeze-excite blocks and LSTMs act like soft switches on the input channels (for squeeze-excite) or on the output-gate's activation vector (for LSTMs). In contrast, the computation in the layers we will introduce provide a quadratic (or L_1) rather than soft-switch response to the values in the spatial context.

Kernel-based convolutional networks [9, 10] also use non-linear combinations of local values. Like classical SVMs [11], kernel-based convolutions use non-linear functions to caste the decision boundary into a higher dimensional space allowing for increased separation between classes. This introduces numerous parameters into the training process, since each support vector is learned (instead of being taken from the training set as was done with classical SVMs). In contrast, our approach uses classical statistical measures, giving each network direct access to a measure of per-channel spatial variability. The only parameters added by our approach are those used to mix these per-channel statistics with the regular linear convolution (or fully-connected) outputs.

For our exploration, we use ResNet [3, 4] (Figure 1), a well-studied DNN architecture, on ImageNet data [12]. The Resnet network starts with a stem, increasing the channel depth from 3 (RGB) to 64 channels. This then feeds into 4 stages, each stage starting with a "projection block" (Figure 2-a) that reduces the spatial resolution by 2 as well as (typically) increasing the number of channels by a factor of 2, followed by "identity blocks" (Figure 2-b) which output the same *shape* (spatial resolution and channels) as was input. Both block types employ two paths, main and shortcut, that branch at the start of the block and are element-wise added at the end of block. In the projection blocks (Figure 2-a), the shortcut path includes a convolution layer that changes the spatial and channel sizes to match the block's output. In the identity blocks (Figure 2-b), the shortcut path is simply an identity path. As shown in Figure 3, the main path of each block has 2 or 3 convolutional layers (depending on the ResNet size). The output channels are each fed into a global average pool which is then fed into the final fully connected classification layer.

3. NON-LINEAR CONTEXT

We explore two different types of non-linear context. Next, we introduce a non-linear, translation-invariant measure of context for use with Conv2D layers. In Section 3.2, we introduce global non-linear pooling that can be used in conjunction with global average pooling.

3.1. *NLConv*: Non-Linear Convolution Layers

With NLConv, our goal is to create a layer that provides local non-linear context, in the same way that an $N \times N$ depth-separable con-

Table 1. Non-linear local metrics tested in NLConv

	(a) centered local mean	(b) sliding local mean
(1) variance	$\sigma^2(x,i,j) = \text{ReLU}(\Delta(x^2,i,j))$ $= \text{ReLU}(m(x^2,i,j) - y_{i,j}^2)$	$\widetilde{\sigma}^2(x,i,j) = \text{ReLU}(m(x_0^2,i,j))$
(2) standard deviation	$\sigma(x,i,j) = \sqrt{\sigma^2(x,i,j)}$	$\widetilde{\sigma}(x_{i,j}) = \sqrt{\tilde{\sigma}^2(x,i,j)}$
(3) L_1 to mean	$\text{L}_1(x_{i,j}) = \Delta(\lvert x\rvert,i,j)$	$\widetilde{\text{L}}_1(x_{i,j}) = \text{ReLU}(m(\lvert x_0\rvert,i,j)$

- $m(f(x),i,j) = \frac{1}{N^2}\sum_{k=-(N-1)/2}^{(N-1)/2}\sum_{l=-(N-1)/2}^{(N-1)/2} f(x_{i+k,j+l})$ is the local mean of $f(x)$ centered at i,j.
- $\Delta(f(x),i,j) = \frac{1}{N^2}\sum_{k=-(N-1)/2}^{(N-1)/2}\sum_{l=-(N-1)/2}^{(N-1)/2} f(x_{i+k,j+l} - y_{i,j})$ is the local mean of $f(x_{i+k,j+l} - y_{i,j})$. Note that the local mean is subtracted before the function $f()$ is applied. Also, note that when the index to x changes within the window, the location used for the local mean y does not change, making it a "centered" local mean.
- $y_{i,j} = m(x,i,j)$ is the local mean for x centered at i,j
- $x_0(i,j) = x_{i,j} - y_{i,j}$ is the difference between x and the local mean for x as seen at i,j. Note that when the index to x_0 changes, the location used for the local mean y also changes, making it a "sliding" local mean.
- N is the size of the local averaging window.
- ReLU is the usual half-wave rectifier. We include the ReLU to avoid errors in backprop.

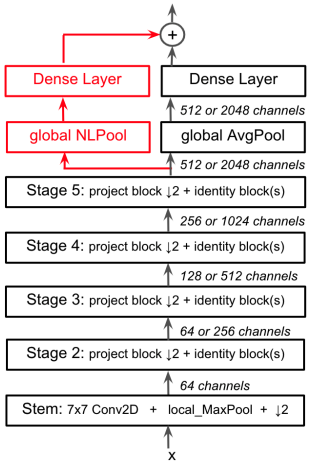

Fig. 1. ResNet architectures.
All ResNets share the same general structure but vary the numbers of repetitions and channels within each stage. ResNet-18 and -34 have fewer channels than ResNet-50 and above. ResNet-18 has only two blocks in each stage (one projection and one identity — see Figure 2) while the larger networks include more identity blocks within each stage. The red portion is the computation that is added in our experiments with global NLPool.

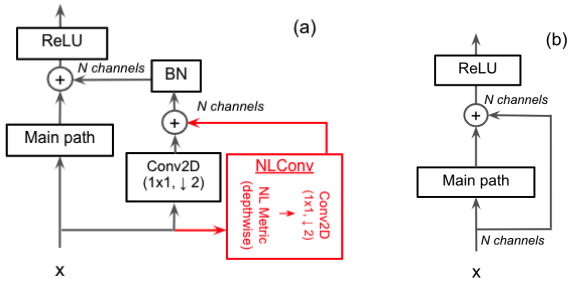

Fig. 2. ResNets use (a) projection and (b) identity blocks.
In (a), the Conv2Ds marked with "↓ 2" downsample by two in x and y. The red portions are the computation that is added in those experiments that use NLConv on the shortcut path of the projection blocks.

volution [1] provides local linear context. We do this by first computing depthwise non-linear metrics and then mixing using a depth-only (1×1) convolution. For the non-linear metrics, we experiment with three simple statistical measures: standard deviation, variance and L_1 distance, see Table 1. In column (a) of Table 1, the computations use a "centered" local mean, centered in the middle of the accumulation window. Eq. 1-a and 2-a are local variance and standard deviation and Eq. 3-a uses a similarly centered local mean as its reference. Column (b) uses a "sliding" local mean, that slides around the accumulation window. This simplifies the computation in Eq. 3-b, allowing us to first compute x_0 and then use local non-linear pooling to determine our non-linear context values. However, this also results in the computation having a larger effective context;

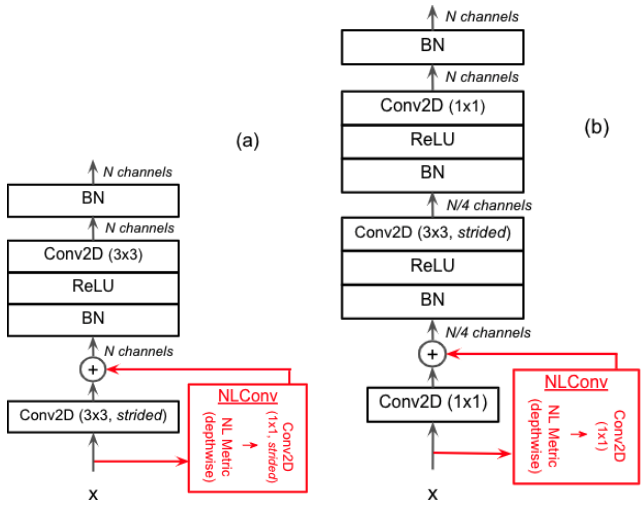

Fig. 3. ResNet main paths are either (a) two (ResNet-18 and -34) or (b) three convolutional-layers deep.
For the Conv2Ds that are marked as "strided", these downsample by two spatially when used in projection blocks but not when used in identity blocks. In both (a) and (b), the red portions are the computation that is added in those experiments that use NLConv on the main paths.

the value at i,j depends on x over a $(2N-1) \times (2N-1)$ window due to the sliding $N \times N$ mean computation.

These computations are all depthwise and are followed by a 1×1 Conv2D (with the appropriate strides), similar to a depthwise convolution, to form an NLConv layer (Figure 3). We found that in ResNet, the most pronounced improvement comes from adding the NLConv in parallel to the first Conv2D of each block (this will be further discussed in Section 4.1). The outputs of the parallel layers are then elementwise added.

3.2. *Global NLPool*: Non-Linear Global Pooling

In ResNet, the spatial outputs of the final stage are collapsed into a single value per channel using global average pooling (Figure 1, top). These average values are then combined using a fully-connected layer to yield the final classification logits. We create a non-linear equivalent to global average pooling, "global NLPool," which uses either variance, standard deviation, or L_1 distance (column (a) of Table 1 with N equal to the spatial dimension of the final layer and sampled only once at the spatial center).

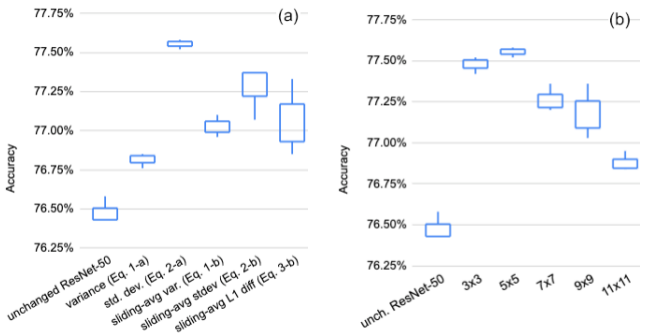

Fig. 4. Parameter exploration for NLConv within ResNet-50.

In our experiments, we place the global NLPool followed by a fully connected layer (Figure 1, in red), in parallel with the global average pool and its following fully connected layer and simply add the two results. The only learnable parameters added by the addition of NLPool are those in the final layer. Because the final layer is fully connected, this increases the number of trainable parameters by 2%-8% (depending on ResNet architecture). Nonetheless, because of the simplicity of fully-connected computations, the inference-time computational penalty is far below 1%.

4. EXPERIMENTS

We begin our experiments with the ResNet-50 network. The network was trained and applied to 224×224 ImageNet inputs. For training, we used SGD (momentum 0.9) on a global batch size of 4096, split across 32 TPUs, using Tensorflow [13]. We used a standard training regime: the learning rate used a linear ramp-up (up to 1.9 across 5 epochs), then a stepwise decay of a factor of 10 each 30 epochs.

All results in this section are based on the performance of three separate training runs. In Figure 4, these results are summarized using a "candlestick": the single line goes from the worst to the best of these 3 runs and the box goes from the average of the bottom two runs to the average of the top two runs. In Table 2, the result is shown as the mean error rate and its variability, across three separate training runs. In Figure 5, due to the complexity of the figure, we simply mark the mean accuracy for three separate training runs.

Figure 4-a shows our results when using the different non-linear measures (Table 1).[1] Local standard deviation (Eq. 2-a in Table 1) clearly outperformed the other non-linear metrics. It also showed significantly less training variability across runs. Due to space constraints, we will report the remainder of our experiments using this standard deviation with a centered local mean.

Figure 4-b shows our results using the standard deviation (Eq. 2-a) with different-sized contexts (values of N). Somewhat surprisingly, both 3×3 and 5×5 contexts clearly outperformed larger context windows. Since the 5×5 did slightly better than 3×3 across all the different training runs, we use that context size for our subsequent NLConv experiments. Understanding why larger contexts did not further improve performance is open for future study.

The global NLPool reduces the entire spatial dimensions on each channel. In our studies across different sized ResNets with global NLPool, using standard deviation gave a more consistent error-rate reduction than either L_1 or variance: both L_1 and variance had at least one run at each size network that failed to improve over the baseline. Therefore, we only report on global NLPool using standard deviation going forward.

Table 2 shows the performance improvement on a large number of ResNet variants. Row (a) lists the error rate for the unchanged network, along with the amount of computation required at inference and number of parameters learned during training. Row (b) gives the performance of ResNet with the additional global NLPool layer. Row (c) adds an NLConv layer (5×5 context) in parallel with the first Conv2D layer of each block's main path (Figure 3, in red) as well as to the projection block shortcuts (Figure 2, in red). Row (d) uses both NLConv and global NLPool.

The most salient feature in Table 2 is that adding non-linear context uniformly improves the accuracy of the unchanged ResNet architecture, up to size 101, with no changes to the training process. Looking in more detail, we see that global NLPool provided the most consistent accuracy improvement; the reduction in the mean error ranged from $\sim$2% (for ResNet-50 and ResNet-101) to $\sim$4% (for ResNet-18 and ResNet-34). Interestingly, the 2–4% accuracy improvement is obtained with under 1% increase in inference-time *computation*. But why does the accuracy improve so much? We suspect the global average pool on the 49 samples per channel (that is, the 7×7 grid which feeds into the average pool) results in enough information loss that even the large numbers of channels cannot fully recover the information. Expanding the network to include standard deviation here provides supplementary information about the within-channel distributions. To ensure that it is not simply the number of extra parameters that provided the benefits, we tested expanding the inputs to this layer by the same amount using finer grain linear pools. This did not improve accuracy; therefore, it is not simply the increase in the parameters that accounts for the improvement. Rather, the non-linear combination across spatial samples in the global NLPool provides useful information to the final layer.

The usefulness of NLConv is more nuanced. For ResNet-18, ResNet-34, and ResNet-50, NLConv consistently improve performance, with an associated increase in inference-time computation of only 8–10% (for ResNet-18 and -34). However, when the network is as large as ResNet-101, NLConv has little effect (though NLPool still has a significant improvement). We speculate that due to the breadth of the network, there are already numerous pathways to convey the equivalent of the NLConv statistics. In Subsection 4.2, we conduct a finer-grained evaluation of which instances of the NLConv provide the most improvement vs. computational cost.

4.1. Alternatives

In addition to the NLConv locations shown in Figures 2 and 3, we report on 2 alternatives: adding an NLConv in parallel with the 7×7 convolution in the stem and adding NLConv to all the main-path convolutions (not just the first).

We tried adding a NLConv in parallel with the 7×7 convolution in the stem of ResNet-50 (the bottom block in Figure 1) and adding the outputs back to the original convolution outputs before the stem's local MaxPool. The advantage of this approach is that non-linear context here results in only a minuscule computation increase (0.04%). We experimented with standard deviations sizes from 3×3 up to 15×15, across 3 independent trials per setting. In all cases, the results were virtually indistinguishable from the unmodified ResNet results. In the stem, there is a massive increase in the representational size as the data goes from an image ($224 \times 224 \times 3 \approx$ 151K values) to $112 \times 112 \times 64 \approx 803$K values. The full image can be *exactly* recreated more than 5 times over in an unmodified ResNet. Given this large information flow, it is not surprising that summary

[1] Figure 4-a does not show the Eq. 3-a due to the difficulty in implementing that function within Tensorflow Keras layers.

Table 2. Error rates across ResNet sizes with non-linear context

architecture	mean error rate ± half range (% improvement)		MACs	trainable parameters
ResNet-18				
a unchanged	30.18% ±0.09%		3.14 B	11.98 M
b with final NLPool	28.81% ±0.08%	(4.5%)	3.14 B	12.19 M
c with NLConv	29.04% ±0.03%	(3.8%)	3.45 B	12.38 M
d with both	28.66% ±0.18%	(5.0%)	3.45 B	12.90 M
ResNet-34				
a unchanged	26.42% ±0.06%		4.99 B	21.98 M
b with final NLPool	25.59% ±0.05%	(3.1%)	4.99 B	22.30 M
c with NLConv	25.81% ±0.16%	(2.3%)	5.41 B	23.05 M
d with both	25.43% ±0.14%	(3.7%)	5.41 B	23.56 M
ResNet-50				
a unchanged	23.52% ±0.07%		5.89 B	25.50 M
b with final NLPool	23.12% ±0.31%	(1.9%)	5.89 B	27.55 M
c with NLConv	22.45% ±0.03%	(4.5%)	8.10 B	32.60 M
d with both	22.38% ±0.42%	(4.8%)	8.10 B	34.65 M
ResNet-101				
a unchanged	21.44% ±0.09		9.60 B	44.44 M
b with final NLPool	20.99% ±0.14	(2.1%)	9.61 B	46.49 M
c with NLConv	21.46% ±0.07	(0.0%)	12.69 B	56.00 M
d with both	21.30% ±0.05	(0.7%)	12.69 B	58.05 M

statistics are not needed.

We also added NLConv to every convolution on the main path (Figure 3), not just the first, while retaining the projection-block shortcut paths. This did result in a slight reduction in the classification error rate: from 22.45% ± 0.03% (Table 2, ResNet-50, line c) down to 22.37% ± 0.08%. However, the improvement was not statistically significant.

4.2. Ablations

In this section, we examine the relative effectiveness of NLConvs at different locations within ResNet-50. Figure 5-a shows an expansive set of results trading off accuracy and computation as we move from an unmodified ResNet-50 to one that has NLConvs on 1, 2, 3 and 4 (**all**) stages of the network. For reference, the plot includes a line between the unmodified ResNet-50 and the ResNet-50 with all NLConvs. The higher the point above the line, the more beneficial it is to keep. Importantly, note that all of the configurations that provide this extra increase in accuracy *include NLConvs in stage 4*. Stage 4 has the largest representational pressure/loss and, consistent with earlier observations, this is the most effective place to use non-linearities.

Figure 5-b shows the change in accuracy and computation as we add or remove NLConvs from different block *types* (projection or identity) and from the two paths on the projection block (main path or shortcut). The best tradeoffs in accuracy vs. computation occur in configurations where NLConv is omitted from all shortcut paths and when it is omitted from projection blocks (point marked "only identity blocks").

5. CONCLUSIONS

Adding non-linear context to convolutional blocks improved the accuracy of ResNet-50 for all combinations of standard deviation, variance, and L_1 distance, at a variety of context sizes, using either centered or sliding local means. Standard deviation with a centered local mean and a 5×5 context was most effective, as was augmenting only the first convolution on the main path of each block.

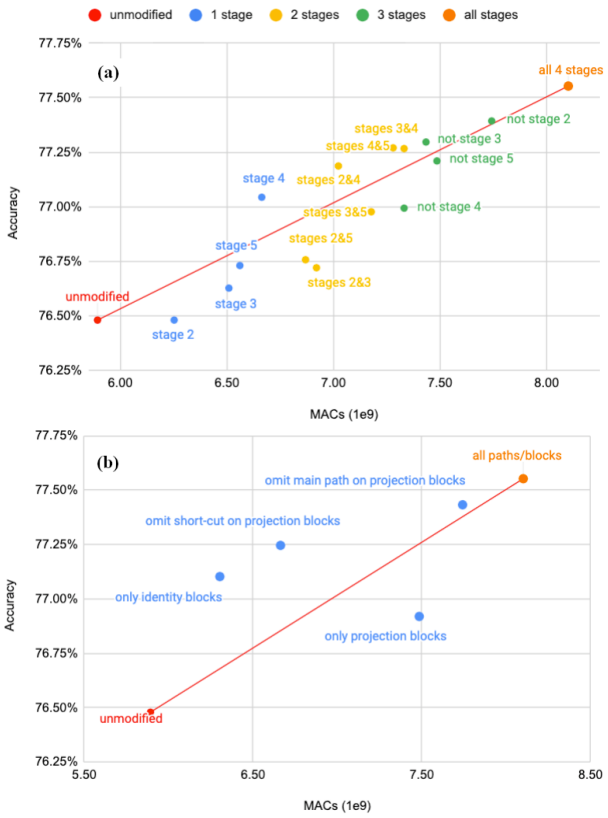

Fig. 5. Ablations for NLConv within ResNet-50.
Examining the effectiveness of NLConv (a) by stage and (b) by block type.

Global NLPool is used in parallel to the standard global average pool. It extends the context of the computations by calculating the standard-deviation of the spatial samples. All sizes of ResNet showed significant reductions in error rates when global NLPool was added, with minimal increases in inference-time computation. The effectiveness of global NLPool at this stage in the network is likely related to the massive reduction in the size of the representation at that layer (by a factor of 49): global NLPool allows otherwise hidden spatial variations to be summarized and passed forward.

NLConv provides local non-linear context where 2D Convolutions are used. Through an expansive set of experiments, we found that NLConv is most effective where the convolutional bottlenecks imposed by a limited number of channels (or resolution) remove useful information. NLConv conveys efficient summaries that simply expanding the width of the network by the same amount did not capture. NLConv, when combined with NLPool, led to 3.7% - 5.0% improvements on ImageNet with ResNet-18, 34 and 50. Both NLConv and NLPool are simple to implement and require no modifications in training algorithm or meta-parameter adjustment.

There are numerous directions for future work, both in alternatives to our approach and the applications addressed. Within our approach, questions remain as to why larger context did not further improve performance. Second, the types of non-linearities employed here were chosen for their simplicity; do other non-linearities, even as simple as higher moments, also provide the same or supplemental benefits? In terms of applications, the extra context provided by our approach may be especially useful in domains in which spatial statistics are exactly what is being modeled – such as video compression and object segmentation.

6. REFERENCES

[1] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[2] Mingxing Tan and Quoc Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds. 09–15 Jun 2019, vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Identity mappings in deep residual networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[5] Shiyu Liang and R. Srikant, "Why deep neural networks for function approximation?," in *2017 International Conference on Learning Representations*, 2017.

[6] Sepp Hochreiter and J urgen Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] Alexey Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *2021 International Conference on Learning Representations*, 2021.

[8] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[9] Hichem Sahbi, "Deep total variation support vector networks," in *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*. 2019, pp. 3028–3038, IEEE.

[10] Hichem Sahbi, "Kernel-based graph convolutional networks," in *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*. 2020, pp. 4887–4894, IEEE.

[11] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[13] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.