Matching Convolutional Neural Networks without Priors about Data

Carlos Eduardo Rosar Kos Lassance*, Jean-Charles Vialatte*† and Vincent Gripon*

*IMT Atlantique
{carlos.rosarkoslassance, jc.vialatte, vincent.gripon}@imt-atlantique.fr
†Cityzen Data

***Abstract*—We propose an extension of Convolutional Neural Networks (CNNs) to graph-structured data, including strided convolutions and data augmentation on graphs. Our method matches the accuracy of state-of-the-art CNNs when applied on images, without any prior about their 2D regular structure. On fMRI data, we obtain a significant gain in accuracy compared with existing graph-based alternatives.**

I. Introduction

Convolutional Neural Networks (CNNs) [1] have been able to surpass traditional machine learning methods in various image based tasks [2], [3]. This is possible as they exploit the learning capabilities of deep neural networks while also taking advantage of the intrinsic regular 2D structure of the data. But when data lacks regular structure [4], there is no natural notion of convolutions, stride/pooling or data augmentation. Such irregularities occur in various domains covering social networks to neuroscience, internet of things, citation graphs, point cloud manifolds... The question of developing solutions that are counterparts of CNNs in irregular domains has recently been a very active field of research.

In this paper we introduce a method that extends CNNs to irregular domains. Contrary to many alternative works, we ensure that our proposed methodology matches the performance of CNNs when applied to regular domains, even without knowledge of the underlying structure. To that end, we infer a graph to represent the topology of the data. From this graph, we infer translations. The weight-sharing schemes of our proposed convolutional layers are then defined based on those translations, as well as data-augmentation and stride.

At the end of the process, the obtained architecture is very similar to a traditional CNN and can thus be trained using the same routines and libraries, and equivalent computational and memory footprints. We first perform experiments on the CIFAR-10 dataset without knowledge about the fact it is made of images. We show that our method is able to reach performance similar to state-of-art CNNs, thus implying that – at least for regular domains – it allows to completely leverage the underlying structure. Then, we perform experiments on an irregular neuroscience dataset and demonstrate a gain in performance compared with completely unstructured deep learning methods and alternative graph-based CNNs.

II. Related Work

Deep learning on graphs can refer to three distinct problems: classification of graphs, of nodes in a graph, or of signals on graphs. In this paper, we are interested only in the latter task that is to leverage the graph structure of signals in deep learning models, by redefining the convolutional layer. Such methods have already been proposed in the literature. We distinguish two categories of solutions.

In the first category, convolution is defined as pointwise multiplication in the spectral domain of the graph, which is defined using the Laplace-Beltrami operator [5]. This method originated the first spectral graph CNNs [6], [7]. An approximation of the spectral graph convolution using Chebychev polynomials has been proposed [8], and has the advantage to be both faster and localized in the vertex domain. Another variant with Cayley polynomials [9] also localizes the convoluted filter in the spectral domain.

In the second category, convolution is defined directly in the vertex domain. These works were originally motivated by chemistry datasets [10], [11]. Convolution is defined as a function of the kernel weights and neighboring vertices (the receptive field), usually based on dot products. As such, it retains the property of being localized and of sharing weights. But there remains the need to specify how the shared weights are allocated in this receptive field [12]. This allocation can depend on an arbitrary order [13], on the number of hops [14], [15], on both vertices and their neighbors [16], [17], on another learned kernel [18], on an attention mechanism [19], on pattern identification [20], or on translation identification [21]. All these methods also differ in the function that maps the receptive fields and the weight kernel to the neuron's outputs. But in the end, these definitions overlap. That is why some authors have proposed unified frameworks [22].

We tackle another point. Given a dataset with no structure between the features of the input vectors, our goal is to demonstrate that we can still define meaningful convolutional, stride/pooling layers. The first step to determine whether the results obtained on unstructured data is satisfactory or not is to stress it on regular data while disregarding its structure. We deal with this step on image datasets. Even though some previous works match the performance of CNNs on image datasets, ours is the first that can do it without structure prior.

III. Methodology

Our method is based on [21], where the authors have introduced a way to infer a graph from training signals, then translations from the obtained graph to design ad-hoc CNNs. We extend this approach and design strided convolutions along graph downscaling, data augmentation and convolutions on downscaled graphs. Figure 1 depicts the proposed method.

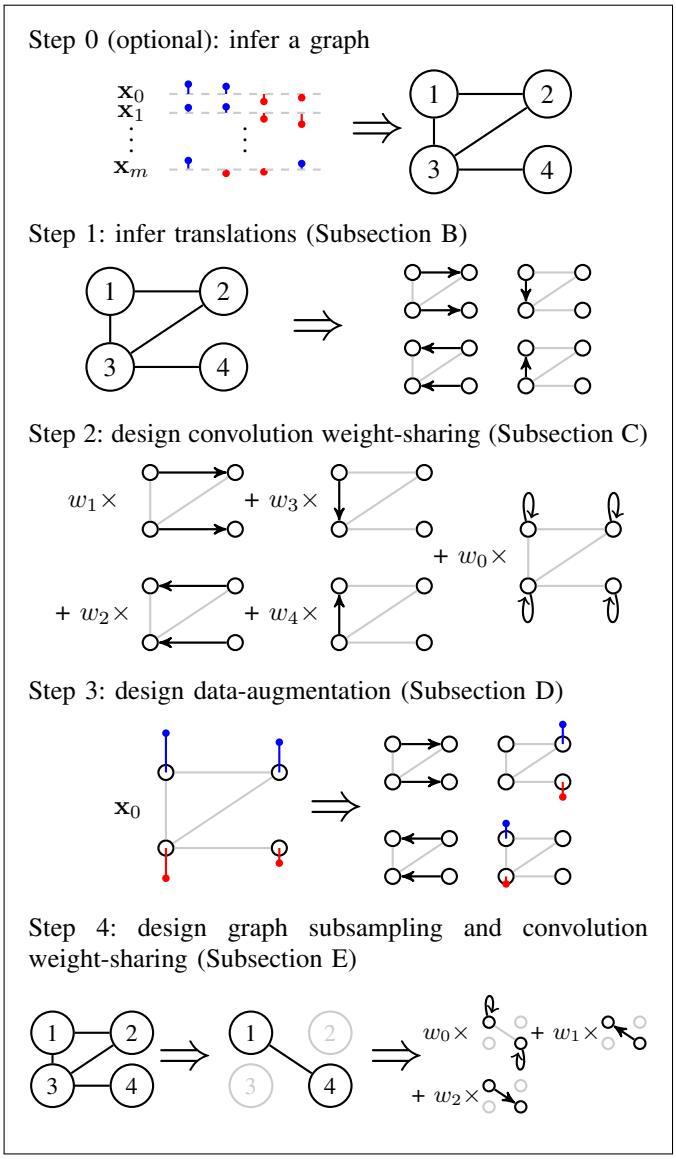

Figure 1. Outline of the proposed method

A. Background

Define a graph $G = \langle V, E \rangle$ with V the set of vertices, and $E \subseteq \binom{V}{2}$ the set of edges. We suppose the graph is connected, as conversely the process can be applied to each connected component of G. We denote by d the max degree of the graph and $n = |V|$ the number of vertices.

The authors of [21] propose to inductively define translations as functions from vertices to vertices as follows:

Definition 1: **Candidate-translation**
A *candidate-translation* is a function $\phi : U \to V$, where $U \subset V$ and such that:

- ϕ is *injective*:
  $\forall v, v' \in U, \phi(v) = \phi(v') \Rightarrow v = v'$,
- ϕ is *edge-constrained*:
  $\forall v \in U, (v, \phi(v)) \in E$,
- ϕ is *strongly neighborhood-preserving*:
  $\forall v, v' \in U, (v, v') \in E \Leftrightarrow (\phi(v), \phi(v')) \in E$.

The cardinal $|V - U|$ is called the *loss* of ϕ. Two candidate-translations ϕ and ϕ' are said to be *aligned* if $\exists v \in V, \phi(v) = \phi'(v)$. We define $N_r(v)$ as the set of vertices that are at most r-hop away from a vertex $v \in V$.

Definition 2: **Translation**
A *translation* in a graph G is a candidate-translation such that there is no aligned translation with a strictly smaller loss, or is the identity function.

Note that if the graph is a 2D grid, obtained translations are exactly natural translations on images [23].

Definition 3: **Local translation**
A *local translation* of center $v \in V$ is a translation in the subgraph of G induced by $N_2(v)$, that has v in its definition domain.

As local translations can't be used to design data augmentation and convolutions on downscaled graphs, we also design proxies to global translations.

Definition 4: **Proxy-translations**
A family of *proxy-translations* $(\psi_p)_{p=0,..\kappa-1}$ initialized by $v_0 \in V$ is defined algorithmically as follows:

1) We place an indexing kernel on $N_1(v_0)$ i.e.
   $N_1(v_0) = \{v_0, v_1, ..., v_{\kappa-1}\}$ with $\forall p, \psi_p(v_0) = v_p$,
2) We move this kernel using each local translation ϕ of center v_0: $\forall p, \psi_p(\phi(v_0)) = \phi(v_p)$,
3) We repeat 2) from each new center reached until saturation. If a center is being reached again, we keep the indexing that minimizes the sum of losses of the local translations that has lead to it.

B. Efficiently Finding Translations

Finding translations is an NP-complete problem [24], such that for large graphs the method is not suitable. In order to break down complexity, the authors of [21] propose to search for local translations. They also introduce approximate translations which we omit for the sake of simplicity, but the description would be similar. We describe in three steps how we efficiently find proxy-translations.

First step: finding local translations
For each vertex $v \in G$, we identify all local translations using a bruteforce algorithm. This process requires finding

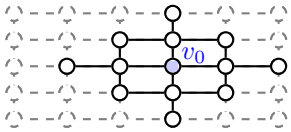

Figure 2. Grid graph (in dashed grey) and the subgraph induced by $N_2(v_0)$ (in black).

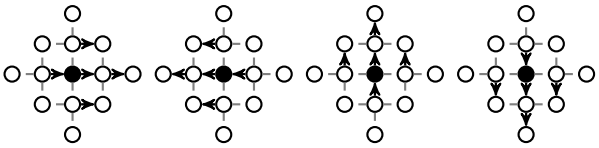

Figure 3. Translations (black arrows) in the induced subgraph (dashed grey) around v_0 (filled in black) that contains v_0 and only some of its neighbors.

all translations in all induced subgraphs. There are n such subgraphs, each one contains at most d local translations. Finding a translation can be performed by looking at all possible injections from 1-hop vertices around the central vertex to any vertex that is at most 2-hops away. We conclude that it requires at most $\mathcal{O}(ndd^{2(d+1)})$ elementary operations and is thus linear with the order of the graph. On the other hand, it suggests that sparsity of the graph is a key criterion in order to maintain the complexity reasonable.

Figure 2 depicts an example of a grid graph and the induced subgraph around vertex v_0. Figure 3 depicts all obtained translations in the induced subgraph.

Second step: using local translations to move a small localized kernel around G

Given an arbitrary[1] vertex $v_0 \in V$, we place an indexing kernel on $N_1(v_0)$ i.e. $N_1(v_0) = \{v_0, v_1, ..., v_{\kappa-1}\}$. Then we move it using every local translations of center v_0, repeating this process for each center that is reached for the first time. We stop when the kernel has been moved everywhere in the graph. In case of multiple paths leading to the same destination, we keep the indexing that minimizes the sum of loss of the series of local translations. We henceforth obtain an indexing of at most κ objects of $N_1(v)$ for every $v \in V$.

This process is depicted in Figure 4. Since it requires moving the kernel everywhere, its complexity is $\mathcal{O}(nd^2)$.

Final step: identifying proxy-translations in G

Finally, by looking at the indexings obtained in the previous step, we obtain a family of proxy-translations defined globally on G. More precisely, each index defines its own proxy-translation. Note that they are not translations because only the local properties have been propagated through the second step, so there can exist aligned candidates with smaller losses. Because of the constraint to keep the paths with the minimum sum of losses, they are good proxies to translations on G.

[1]In practice we run several experiments while changing the initial vertex and keep the best obtained result.

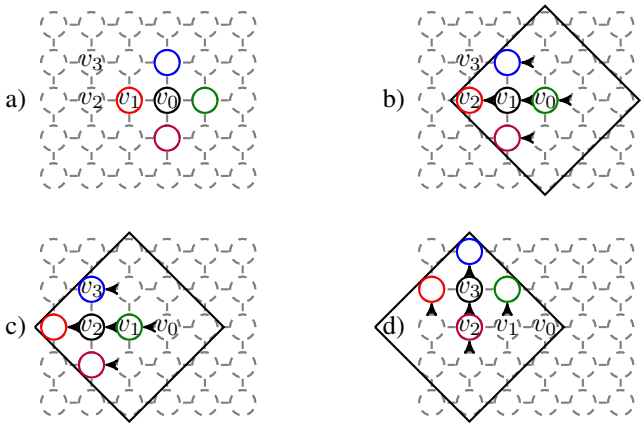

Figure 4. Illustration of the translation of a small indexing kernel using translations in each induced subgraph. Kernel is initialized around v_0 (a), then moved left around v_1 (b) using the induced subgraph around v_0, then moved left again around v_2 (c) using the induced subgraph around v_1 then moved up around v_3 (d) using the induced subgraph around v_2. At the end of the process, the kernel has been localized around each vertex in the graph.

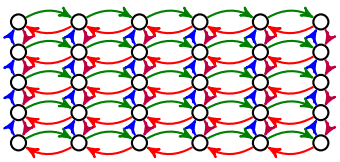

Figure 5. Proxy-translations in G obtained after moving the small kernel around each vertex. Each color corresponds to one translation.

An illustration on a grid graph is given in Figure 5. The complexity is $\mathcal{O}(nd)$. Overall, all three steps are linear in n.

C. Extended Convolution Layers

Let $(\psi_p)_{p=0,..,\kappa-1}$ be the proxy-translations identified on G with the convention that $\psi_0 = id$ is the identity function, and where κ is the number of weights in the indexing kernel.

The operation of the *extended convolution layer* centered on the vertex $v \in V$ is defined as:

$$\mathbf{y}_v = h\left(\sum_{p=0}^{\kappa-1} w_p \mathbf{x}_{\phi_p(v)} + b\right)$$

where h is the activation function, b is the bias term, $\mathbf{x}_\perp = 0$ and:

$$\begin{cases} \phi_p(v) = \psi_p(v) & \text{if } \psi_p \text{ is defined on } v \\ \phi_p(v) = \perp \notin V & \text{else} \end{cases}.$$

Note that we defined convolution layers using the formalism of proxy-translations, but they can also be defined using only the formalism of local translations [21].

D. Extended Data Augmentation

Once translations are obtained on G, one can use them to move training vectors, artificially creating new ones. Note that this type of data-augmentation is poorer than for images since no flipping, scaling or rotations are used.

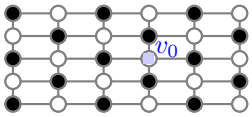

Figure 6. Downscaling of the grid graph. Disregarded vertices are filled in.

E. Extended Downscaling Layers

Downscaling is a tricky part of the process because it supposes one can somehow regularly sample vectors. As a matter of fact, a nonregular sampling is likely to produce a highly irregular downscaled graph, on which looking for translations irremediably leads to poor accuracy, as we noticed in our experiments.

We rather define the translations of the strided graph using the previously found proxy-translations on G.

First step: extended convolution with stride r

Given an arbitrary initial vertex $v_0 \in V$, the set of kept vertices $V_{\downarrow r}$ is defined inductively as follows:

- $V_{\downarrow r}^0 = \{v_0\}$,
- $\forall t \in \mathbb{N}, V_{\downarrow r}^{t+1} = V_{\downarrow r}^t \cup \{v \in V, \forall v' \in V_{\downarrow}^t, v \notin N_{r-1}(v') \wedge \exists v' \in V_{\downarrow r}^t, v \in N_r(v')\}$.

This sequence is nondecreasing and bounded by V, so it eventually becomes stationary and we obtain $V_{\downarrow r} = \lim_t V_{\downarrow r}^t$. Figure 6 illustrate the first downscaling $V_{\downarrow 2}$ on a grid graph.

The output neurons of the extended convolution layer with stride r are $V_{\downarrow r}$.

Second step: convolutions for the strided graph

Using the proxy-translations on G, we move a localized r-hop indexing kernel over G. At each location, we associate the vertices of $V_{\downarrow r}$ with indices of the kernel, thus obtaining what we define as induced $_{\downarrow r}$-translations on the set $V_{\downarrow r}$. In other words, when the kernel is centered on v_0, if $v_1 \in V_{\downarrow r}$ is associated with the index p_0, we obtain $\phi_{p_0}^{\downarrow r}(v_0) = v_1$. Subsequent convolutions at lower scales are defined using these induced $_{\downarrow r}$-translations similarly to Subsection C.

IV. Experiments

To validate our method we performed experiments with two different datasets, CIFAR-10 [25] and PINES fMRI dataset [26]. The code is available at github.com/brain-bzh/MCNN.

A. CIFAR-10

On the CIFAR-10 dataset, our models are based on a variant of a deep residual network, namely PreActResNet18[2]. We tested different combinations of graph support and data augmentation. For the graph support, we use either a regular 2D grid or either an inferred graph obtained by keeping the four neighbours that covary the most. Table I summarizes out results. In particular, it is interesting to note that results obtained without any structure prior (91.07%) are only 2.7% away from the baseline using classical CNNs on images (93.80%). This gap is even smaller (less than 1%) when using the grid prior. Also, without priors our method significantly outperforms the others.

Table I
CIFAR-10 result comparison table.

Support	MLP [27]	CNN	Grid Graph (Given)		Covariance Graph (Inferred)
			[8]	Proposed	Proposed
Full Data Augmentation	78.62%	**93.80%**	85.13%	93.94%	92.57%
Data Augmentation - Flip	——	92.73%	84.41%	92.94%	91.29%
Graph Data Augmentation	——	92.10%[a]	—	92.81%	**91.07%**[b]
None	69.62%[b]	87.78%	—	88.83%	85.88%[b]

[a] As the CNN does not have a graph support we used the covariance graph as support for the graph data augmentation.
[b] No priors about the structure

B. PINES fMRI

The PINES dataset consists of fMRI scans on 182 subjects, during an emotional picture rating task[26]. We fetched individual first-level statistical maps (beta images) for the minimal and maximal ratings from https://neurovault.org/collections/1964/, to generate the dataset. Full brain data was masked on the MNI template and resampled to a 16mm cubic grid, in order to reduce dimensionality of the dataset while keeping a regular geometrical structure. Final volumes used for classification contain 369 signals for each subject and rating.

We used a shallow network. The results on Table II show that our method was able to improve over CNNs, MLPs and other graph-based extended convolutional neural networks.

Table II
PINES fMRI dataset accuracy comparison table.

Graph	None		Neighborhood Graph	
Method	MLP	CNN (kernel 1x1)	[8]	Proposed
Accuracy	82.62%	84.30%	82.80%	**85.08%**

V. Conclusion

We proposed a new methodology that extends classical convolutional neural networks to irregular domains represented by a graph. The methodology scales linearly well with the order of the graph. Moreover, training can be performed using existing libraries for deep learning.

We performed experiments and showed that our method is able to match performance of classical convolutional neural networks on images without explicit knowledge about the underlying regular 2D structure. It also significantly outperforms existing extended convolutional neural networks alternatives based on graphs. We also demonstrated the ability of the method to adapt to slightly irregular domains by performing experiments on a neuroimage dataset.

However, the main limitation is that on highly irregular domains, the obtained translations aren't very helpful to design meaningful convolutions, especially if the degree of the graph varies a lot. Hence this requires to add constraints to the graph inferring step to obtain an exploitable graph if it is not.

Future work includes extending to highly irregular domains, which might require to revisit the definitions of translations.

Acknowledgements

This work was funded in part with the support of Région Bretagne and computations were performed with the use of Nvidia GPUs, courtesy of Nvidia.

REFERENCES

[1] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[4] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[5] F. R. K. Chung, *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, 1996.

[6] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[7] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[8] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3837–3845.

[9] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *arXiv preprint arXiv:1705.07664*, 2017.

[10] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems*, 2015, pp. 2215–2223.

[11] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of computer-aided molecular design*, vol. 30, no. 8, pp. 595–608, 2016.

[12] J.-C. Vialatte, V. Gripon, and G. Mercier, "Generalizing the convolution operator to extend cnns to irregular domains," *arXiv preprint arXiv:1606.01166*, 2016.

[13] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, 2016, pp. 2014–2023.

[14] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1993–2001.

[15] J. Du, S. Zhang, G. Wu, J. M. Moura, and S. Kar, "Topology adaptive graph convolutional networks," *arXiv preprint arXiv:1710.10370*, 2017.

[16] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," *arXiv preprint arXiv:1611.08402*, 2016.

[17] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," *arXiv preprint arXiv:1704.02901*, 2017.

[18] J.-C. Vialatte, V. Gripon, and G. Coppin, "Learning local receptive fields and their weight sharing scheme on graphs," *arXiv preprint arXiv:1706.02684*, 2017.

[19] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *stat*, vol. 1050, p. 20, 2017.

[20] A. Sankar, X. Zhang, and K. C.-C. Chang, "Motif-based convolutional neural network on graphs," *arXiv preprint arXiv:1711.05697*, 2017.

[21] B. Pasdeloup, V. Gripon, J.-C. Vialatte, and D. Pastor, "Convolutional neural networks on irregular domains through approximate translations on inferred graphs," *arXiv preprint arXiv:1710.10035*, 2017.

[22] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

[23] N. Grelier, B. Pasdeloup, J.-C. Vialatte, and V. Gripon, "Neighborhood-preserving translations on graphs," in *Proceedings of IEEE GlobalSIP*, 2016, pp. 410–414.

[24] B. Pasdeloup, V. Gripon, N. Grelier, J.-C. Vialatte, and D. Pastor, "Translations on graphs with neighborhood preservation," *arXiv preprint arXiv:1709.03859*, 2017.

[25] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[26] L. J. Chang, P. J. Gianaros, S. B. Manuck, A. Krishnan, and T. D. Wager, "A sensitive and specific neural signature for picture-induced negative affect," *PLoS biology*, vol. 13, no. 6, p. e1002180, 2015.

[27] Z. Lin, R. Memisevic, and K. R. Konda, "How far can we go without convolution: Improving fully-connected networks," *CoRR*, vol. abs/1511.02580, 2015. [Online]. Available: http://arxiv.org/abs/1511.02580