RECURRENT KNOWLEDGE DISTILLATION

Silvia L. Pintea[1], *Yue Liu*[2], *Jan C. van Gemert*[1]

[1]Vision Lab, Delft University of Technology, Netherlands
[2] School of Electrical Engineering and Computer Science, KTH, Stockholm, Sweden

ABSTRACT

Knowledge distillation compacts deep networks by letting a small student network learn from a large teacher network. The accuracy of knowledge distillation recently benefited from adding residual layers. We propose to reduce the size of the student network even further by recasting multiple residual layers in the teacher network into a single recurrent student layer. We propose three variants of adding recurrent connections into the student network, and show experimentally on CIFAR-10, Scenes and MiniPlaces, that we can reduce the number of parameters at little loss in accuracy.

***Index Terms*—** Knowledge distillation, compacting deep representations for image classification, recurrent layers.

1. INTRODUCTION

Deep learning requires deep computational pockets. Current models for image classification [1, 2], object detection [3, 4], semantic segmentation [5, 6] use millions of parameters. The memory requirements of such large models prevent real-world applicability in limited memory scenarios such as surveillance, home safety devices, industrial robots, autonomous driving, etc. In this paper we investigate low-memory approximations of high-accuracy models.

Knowledge distillation [7, 8] exploits large teacher models to help train student models that are more compact yet retain good accuracy. Recent methods use thin yet deep students [9], reaping the benefits of residual layers [1] to train extremely deep architectures. For training such deep networks, the teacher-student similarity losses are computed at similar depths in the architecture [10, 9], requiring student models to have similar depths as their teachers.

In this paper we propose to further reduce the parameters of deep student networks by sharing weights between residual layers. We are inspired by residual learning seen as an iterative refinement scheme [11], and that multiple residual connections can be seen as an iterative recurrent connection "unrolled" over time [12, 13, 14]. We propose the use of recurrent connections for compacting the information of multiple residual layers of the teacher network into a single recurrent layer in the student model. In Fig 1 we illustrate our approach. We make the following contributions: (i) student memory reduction by using recurrent connections instead of residuals; (ii) exploring three variants of adding recurrence; (iii) experiments on CIFAR-10, Scenes and MiniPlaces show substantially reduced parameters with similar accuracy.

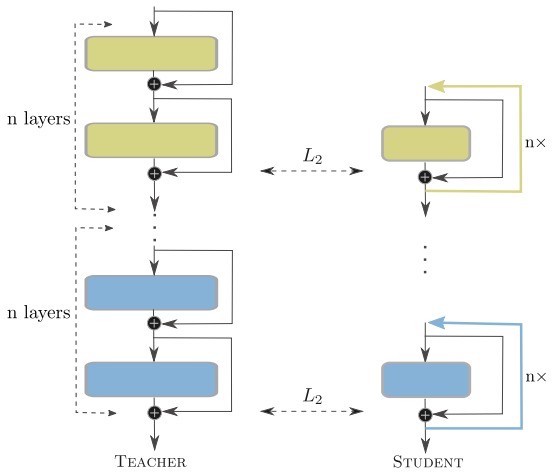
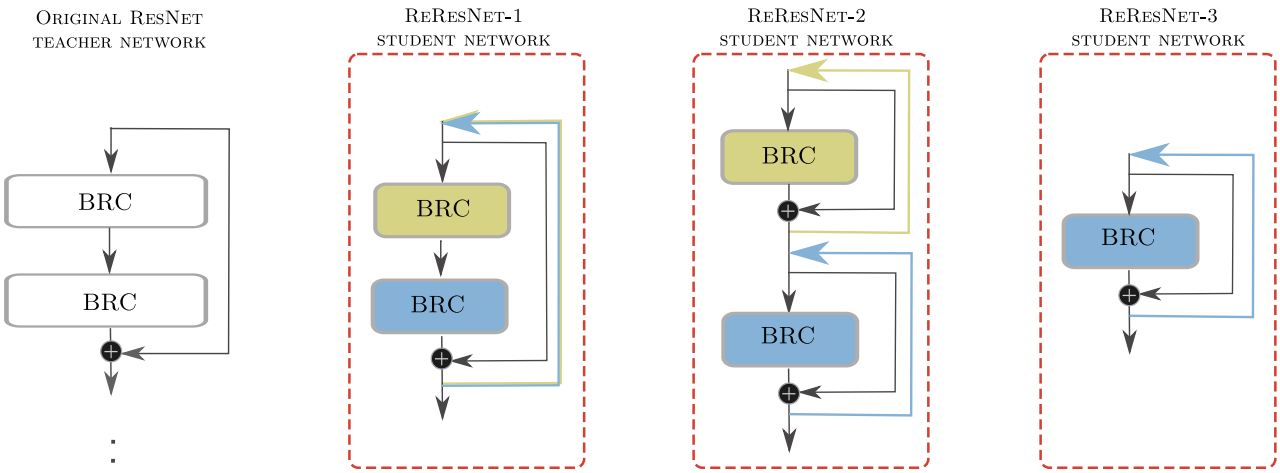


Fig. 1. Multiple residual layers in the teacher network (left) are mapped to a single layer in the student network (right) through recurrence. The L_2 measures similarity between networks, and similar colors correspond to similar layers between networks. Note the reduction in layer parameters.

2. RELATED WORK

Knowledge distillation. Knowledge between multiple modalities [15] can be transferred. Doing the opposite of our aim, [16] trains a deep architecture from a recurrent language model, while [17, 18] improve time and accuracy, rather than reducing parameters. In contrast to these works, we aim at parameter reduction by training a low-parameter student network from a parameter-intensive residual teacher network through the use of recurrent connections.

Time efficiency. Extremely deep networks can be trained efficiently by controlling the information flow through the use of multiple interconnections between layers and gating [19, 20]. While in [21] speedups are obtained by pruning convolutional kernels in a group-wise fashion. The work in [22] uses shallower wider neural networks for training time efficiency. Unlike these methods, we do not focus on improving the speed,



(a) Original. (b) RERESNET-1 (c) RERESNET-2 (d) RERESNET-3

Fig. 2. Our three proposed variants for the student models together with the original block. We display without color the units that are not tied over time and in the same color (yellow or blue) the units that share parameters over time. (a) The original residual blocks, each with two BRC (BN-RELU-Conv) units inside. (b) RERESNET-1: both units are tied over time in an interleaved fashion: the first one (in yellow) is tied at $t \in \{1.., 2n+1\}$, and the second (in blue) at $t \in \{2.., 2n\}$. (c) RERESNET-2: the two BRC units broken apart, each being shared sequentially: the first one (in yellow) is tied at $t \in \{1.., n\}$, and the second (in blue) at $t \in \{n+1.., 2n\}$. (d) RERESNET-3: the two BRC units are collapsed into a single unit, shared at $t \in \{1.., n\}$. Where n denotes the number of recurrence steps.

but on substantially reducing the number of parameters.

Memory efficiency. Network parameters can be compressed through low-rank and sparse decomposition of weight matrices [23, 24]. Alternatively, binarization [25, 26] can reduce memory usage at the expense of a small drop in performance. Others exploit circular matrices in the Fourier domain for projecting the network feature maps for dimensionality reduction [27, 28]. Our method is intended to benefit before such approximations are applied. We train a low-parameter student network to mimic as closely as possible a parameter-intensive teacher network.

3. METHOD

3.1. Recurrent ResNet for knowledge distillation

Given a residual block i, with a corresponding mapping function, $f_i(\cdot)$, and weights $\mathbf{W}_i$, the output of its application, $\hat{f}_i(\cdot)$, to its input feature map, $\mathbf{x}_{i-1}$, is defined [1] as:

$$\mathbf{x}_i = \hat{f}_i(\mathbf{x}_{i-1}, \mathbf{W}_i) = f_i(\mathbf{x}_{i-1}, \mathbf{W}_i) + \mathbf{x}_{i-1}. \quad (1)$$

All the feature maps in one residual block have the same width and height, therefore, the same spatial scale. We define a recurrent unit at time $t > 1$, by sharing the weights at the same spatial scale in the residual block.

$$\mathbf{x}_i(t) = \hat{f}_i(\mathbf{x}_i(t-1), \mathbf{W}_i), \quad (2)$$

where the feature maps $\mathbf{x}_i$, become a function of time, t.

Each residual block of a ResNet architecture [1] consists of two BRC units, composed of: batch normalization (BN), rectified linear units (RELU), and a convolution (Conv). We coin our recurrent residual model: RERESNET. Starting from a residual teacher network [1], we consider all three different possibilities of adding recurrence into the student network, by recurring both BRC units in a residual block. Fig 2 illustrates these three variants of the student model.

ReResNet-1: interleaved parameter tying. The first unit is tied over odd timesteps, $t = 2n + 1$, and therefore its associated weights are shared over these timesteps. The second unit in the block is tied over even timesteps $t = 2n$, where n is the number of recurrence steps considered. See Fig 2(b).

ReResNet-2: sequential parameter tying. The first unit is tied over the timesteps $t = \{1, .., n\}$, and their weights are shared over those timesteps. While the second unit is tied over timesteps $t = \{n+1, .., 2n\}$. See Fig 2(c).

ReResNet-3: single unit with shared weights. Collapsing the two BRC units to only one unit with shared weights which is then recurred over time $t = \{1, .., n\}$. See Fig 2(d).

3.2. Incorporating the teacher-student similarity loss

Our loss $\mathcal{L}(\mathbf{x}, \{\mathbf{W}^s\})$ over the input data $\mathbf{x}$ and network parameters of the student model, $\mathbf{W}^s$, is a combination of the softmax classification loss, $\mathcal{L}_{\text{cls}}$, and the teacher-student similarity loss, $\mathcal{L}_{\text{ts}}$:

$$\mathcal{L}(\mathbf{x}, \{\mathbf{W}^s\}) = \mathcal{L}_{\text{cls}}(\mathbf{x}, \{\mathbf{W}^s\}) + \lambda \mathcal{L}_{\text{ts}}(\mathbf{x}, \{\mathbf{W}^s, \mathbf{W}^t\}), \quad (3)$$

where the weights of the teacher model $\mathbf{W}^t$, are fixed, and λ is the trade off parameter between the two losses. The

teacher-student similarity loss $\mathcal{L}_{\text{ts}}$, is the L_2 distance between the teacher-student activations at a set of layer pairs, $\mathcal{K}$:

$$\sum_{(i,j)\in\mathcal{K}} \left(\frac{\sum_{c=1}^{N} \hat{f}_i^c(\mathbf{x}_i, \mathbf{W}_i^s)^2}{\| \sum_{c=1}^{N} \hat{f}_i^c(\mathbf{x}_i, \mathbf{W}_i^s)^2 \|_2} - \frac{\sum_{c=1}^{M} \hat{f}_j^c(\mathbf{x}_j, \mathbf{W}_j^t)^2}{\| \sum_{c=1}^{M} \hat{f}_j^c(\mathbf{x}_j, \mathbf{W}_j^t)^2 \|_2} \right)^2, \quad (4)$$

where N and M are the number of channels in the student and teacher feature maps, respectively. To allow for different number of channels between the teacher network and the student network, the squared values of the feature maps in the two networks are first accumulated over the channels as in [9], and subsequently, the feature maps are normalized.

3.3. Implementing recurrence and shared gradients

We implement recurrence by sharing the parameters of the convolutional layers, allowing each depth to learn its own BN (Batch Normalization) parameters. We do so, as we have found experimentally there is a considerable drop in performance when sharing also the BN parameters.

Since we can only recur over convolutional layers with the same spatial size, we add one extra convolution between recurrent residual blocks with different spatial sizes. During backpropagation, each shared residual mapping function $\hat{f}_i(\cdot)$ as given in Eq. (2), at each time t, will give an associated gradient for the weights to be learned in that layer, $\partial \mathbf{W}_i^s(t)$. As in [29], we update the shared weights of the student network in one layer, $\mathbf{W}_i^s$, with the sum of all the gradients across all time steps, t.

$$\frac{\partial \mathcal{L}(\mathbf{x}, \{\mathbf{W}^s\})}{\partial \mathbf{W}_i^s} = \sum_{t=1}^{n} \frac{\partial \mathcal{L}(\mathbf{x}, \{\mathbf{W}^s\})}{\partial \mathbf{W}_i^s(t)}, \quad (5)$$

where $\mathcal{L}(\mathbf{x}, \{\mathbf{W}^s\})$ is our final loss as given in Eq. (3), and n is the number of recurrence steps for that recurrent BRC unit.

4. EXPERIMENTAL EVALUATION

4.1. Experimental analysis

For all experiments we standardize the data by translating it to mean zero and scaling it to unit standard deviation. We use a weight decay of $1.0e-5$ and momentum of 0.9 and initialize the weights following [30].

CIFAR-10 [31]. We first evaluate our model on CIFAR-10 using the WRN-18-2 model from [22] with a wide residual teacher model with 18 layers and 3 blocks and make the student networks half as wide by using only half of the number of filters in the convolutional layers. The teacher and student models are trained from scratch using batch size 128. The starting learning rate is 0.1 and is decreased by a factor of 10 every $40K$ iterations and $60K$ iterations. For data augmentation, following [1] we pad 4 pixels on each side, take random crops and add random horizontal flips.

# Recurs	Parameters %	Parameters #	Accuracy
Teacher	100%	1.235 M	93.25%
1	6%	73 K	86.95%
2	6%	73 K	88.17%
3	6%	73 K	88.33%
4	6%	74 K	88.39%
5	6%	74 K	88.24%
6	6%	74 K	87.95%
12	6%	75 K	88.03%

Table 1. **Exp. 1:** The effect of recurrence on performance on CIFAR-10. We test between 1 and 6 repetitions, and 12 repetitions on our lightest student model: RERESNET-3. After 3-4 recurrences performance is stable.

Network	# Recurs	Parameters %	Parameters #	Accuracy
Teacher		100%	1.235 M	93.28%
RERESNET-1	3	10%	122 K	89.81%
RERESNET-2	3	10%	122 K	89.25%
RERESNET-3	3	6%	73 K	88.33%
RERESNET-1	6	10%	124 K	89.99%
RERESNET-2	6	10%	124 K	89.00%
RERESNET-3	6	6%	74 K	87.95%

Table 2. **Exp. 2:** Comparison of RERESNET-1, RERESNET-2 and RERESNET-3 on CIFAR-10 with 3 and 6 recurrences. Overall, RERESNET-1 performs slightly better than the other two variants, however there is not a significant difference in the performance of the three considered models.

Exp. 1: How many times to recur? In table 1 we show the effect of recurrence on accuracy using our smallest model, RERESNET-3. For this experiment we do not use the teacher-student similarity loss. Performance is stable to recurrence. In our subsequent experiments we evaluate our models when using 3 and 6 repetitions in the recurrent links.

Exp. 2: How to add recurrence? In table 2 we compare our three proposed variants RERESNET-1, RERESNET-2, RERESNET-3 on CIFAR-10 when using 3 and 6 recurrences. Here we do not use the teacher-student similarity loss. The RERESNET-1 performs slightly better than the other two variants while having an intuitive manner of adding recurrence: the complete block is recurred, rather than each convolution in the block separately. In the subsequent experiments we use the RERESNET-1 model and refer to it as RERESNET.

Exp. 3: Comparison with existing work. Table 3 shows the accuracy and number of parameters of our RERESNET*, enhanced with the teacher-student similarity loss, when com-

Network	Parameters		Accuracy
	%	#	
Teacher	100%	1.235 M	93.28%
Maxout [32]	>400%	>5 M	90.62%
Prob. maxout [33]	>400%	>5 M	90.61%
FitNet [10]	200%	2.5 M	91.61%
Highway [19]	100%	1.25 M	91.20%
NIN [2]	78%	970 K	91.19%
Circulant CNN [27]	9.7%	120 K	84.29%
RERESNET* (3 recurs)	10%	122 K	90.29%
RERESNET* (6 recurs)	10%	124 K	90.49%

Table 3. **Exp. 3:** Comparison of performance and number of parameters on CIFAR-10 for our RERESNET*, enhanced with the teacher-student similarity loss, when compared to other efficient methods such as FitNets [10] and Circulant CNN [27], and other large architectures such as NIN (Network in network) [2], Highway networks [19], Maxout networks [32], and probabilistic Maxout networks [33]. Our method outperforms the Circulant CNN while having a similar number of parameters, and performs on par with the Maxout architectures [32, 33] while having 40× less parameters.

pared to popular network architectures.

We compare with Circulant CNN [27] which has a similar size with our model, as well as the FitNet proposed in [10]. We additionally compare with other larger models such as NIN (Network in Network) [2], Highway networks [19], Maxout network[32], and probabilistic Maxout network [33]. Our proposed approach outperforms 3 out of these 6 models considered. We outperform the Circulant CNN [27] while having a similar number of parameters, and the two Maxout models [32, 33] while using 40× less parameters.

4.2. Performance on scene recognition

To evaluate the generalization capacity of our method we evaluate the performance of our RERESNET* enhanced with the teacher-student similarity loss, on two scene datasets.

Exp. 4: Scenes [34]. Given the small dataset we use a shallower but wider teacher model: a wide residual net from [22] with width factor 4 and 12 layers, using only 2 residual blocks rather than 3. Given the small dataset size, we extract features from the last residual group of a pre-trained ResNet-50 [1] on ImageNet [35] and train our teacher/student models on top of that. Batch size is set to 128. The learning rates used are 0.01 for the first $2K$ iterations and 0.001 for the rest. Following [36], we resize the input images to 224×224 pixels. Table 4 shows the performance on the Scenes dataset when comparing our teacher performance with the performance of our RERESNET* student architecture. We notice a 3% decrease in performance at the gain of more than $5\times$ reduction in parameters.

Network	Parameters		Accuracy
	%	#	
Teacher	100%	23.64 M	71.80%
RERESNET*	18%	4.15 M	68.73%

Table 4. **Exp. 4:** On the Scenes dataset we lose 3% in accuracy at for a more than 5 × gain in parameter usage.

Network	Parameters		Top-1	Top-5
	%	#		
Teacher	100%	6.07 M	47.54%	76.82%
RERESNET*	33%	1.98 M	47.56%	77.42%

Table 5. **Exp. 5:** On Miniplaces the student performs similar to the teacher while using 3× less parameters.

Exp. 5: MiniPlaces [37]. On MiniPlaces the teacher architecture corresponds to the standard residual network proposed in [1], ResNet-34. Here the student model is as wide as the teacher model. The teacher and student networks are trained from scratch using batch size 256. We start with learning rate 0.1, and we decrease it by 10 every $70K$ and $160K$ iterations. As input during training we use random image crops of 112×112 pixels from the initial 128× 128 images. In table 5 the performance on MiniPlaces is evaluated, where we compare the teacher performance with the performance of our RERESNET* student model. In this experiment we obtain the same performance as our teacher model while using $3\times$ less parameters in our student model.

5. CONCLUSION

In this paper we focus on model parameter reduction by using knowledge distillation for learning compact student models from wider and deeper residual models. Our student models use recurrent connections for compacting the information and allowing for shallow student networks. We propose three variants of our student model, RERESNET-1, RERESNET-2 and RERESNET-3, in which the convolutional layers in the residual blocks are tied over time in different manners. We evaluate our model choices as well as compare with existing work in terms of accuracy and used number of parameters, and show experimentally that our models can achieve comparable performance, using considerably less parameters.

One limitation of our approach is that recurrence is now added simply through sharing parameters over time which lacks in descriptive power. We believe improvements can be obtained by using gating functions as in the case of LSTM [38] blocks, to control the information remembered. The student networks would benefit from this approach.

6. REFERENCES

[1] K He, X Zhang, S Ren, and J Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[2] M Lin, Q Chen, and S Yan, "Network in network," *ICLR*, 2013.

[3] J Dai, Y Li, K He, and J Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *NIPS*, 2016, pp. 379–387.

[4] T Y Lin, P Dollár, R Girshick, K He, B Hariharan, and S Belongie, "Feature pyramid networks for object detection," *CVPR*, 2016.

[5] K He, G Gkioxari, P Dollár, and R Girshick, "Mask r-cnn," *ICCV*, 2017.

[6] E Shelhamer, J Long, and T Darrell, "Fully convolutional networks for semantic segmentation," *PAMI*, vol. 39, no. 4, pp. 640–651, 2017.

[7] G Hinton, O Vinyals, and J Dean, "Distilling the knowledge in a neural network," *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[8] J Ba and R Caruana, "Do deep nets really need to be deep?," in *NIPS*, 2014, pp. 2654–2662.

[9] Sergey Zagoruyko and Nikos Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *ICLR*, 2017.

[10] A Romero, N Ballas, S E Kahou, A Chassang, C Gatta, and Y Bengio, "Fitnets: Hints for thin deep nets," *ICLR*, 2015.

[11] K Greff, R K Srivastava, and J Schmidhuber, "Highway and residual networks learn unrolled iterative estimation," *ICLR*, 2016.

[12] Q Liao and T Poggio, "Bridging the gaps between residual learning, recurrent neural networks and visual cortex," *primates*, vol. 31, pp. 28, 2016.

[13] Boulch, "Sharesnet: reducing residual network parameter number by sharing weights," *CoRR*, 2017.

[14] A Veit, M Wilber, and S Belongie, "Residual networks are exponential ensembles of relatively shallow networks," in *NIPS*, 2016.

[15] S Gupta, J Hoffman, and J Malik, "Cross modal distillation for supervision transfer," in *CVPR*, 2016, pp. 2827–2836.

[16] W Chan, N R Ke, and I Lane, "Transferring knowledge from a rnn to a dnn," *NTERSPEECH*, 2015.

[17] T Chen, I Goodfellow, and J Shlens, "Net2net: Accelerating learning via knowledge transfer," *ICLR*, 2015.

[18] J Yim, D Joo, J Bae, and J Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *CVPR*, 2017, pp. 4133–4141.

[19] R K Srivastava, K Greff, and J Schmidhuber, "Highway networks," *ICML workshop on Deep Learning*, 2015.

[20] R K Srivastava, K Greff, and J Schmidhuber, "Training very deep networks," in *NIPS*, 2015, pp. 2377–2385.

[21] V Lebedev and V Lempitsky, "Fast convnets using group-wise brain damage," in *CVPR*, 2016, pp. 2554–2564.

[22] S Zagoruyko and N Komodakis, "Wide residual networks," *BMVC*, 2016.

[23] X Yu, T Liu, X Wang, and D Tao, "On compressing deep models by low rank and sparse decomposition," in *CVPR*, 2017, pp. 7370–7379.

[24] X Zhang, J Zou, K He, and J Sun, "Accelerating very deep convolutional networks for classification and detection," *PAMI*, vol. 38, no. 10, pp. 1943–1955, 2016.

[25] M Rastegari, V Ordonez, J Redmon, and A Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016, pp. 525–542.

[26] A Daniely, N Lazic, Y Singer, and K Talwar, "Sketching and neural networks," *ICLR workshop*, 2017.

[27] Y Cheng, F X Yu, R S Feris, S Kumar, A Choudhary, and S F Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *ICCV*, 2015, pp. 2857–2865.

[28] Yunhe Wang, Chang Xu, Chao Xu, and Dacheng Tao, "Beyond filters: Compact feature map for portable deep model," in *ICML*, 2017, pp. 3703–3711.

[29] M Liang and X Hu, "Recurrent convolutional neural network for object recognition," in *CVPR*, 2015, pp. 3367–3375.

[30] K He, X Zhang, S Ren, and J Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, 2015, pp. 1026–1034.

[31] A Krizhevsky and G Hinton, "Learning multiple layers of features from tiny images," 2009.

[32] I J Goodfellow, D Warde-Farley, M Mirza, A Courville, and Y Bengio, "Maxout networks," *JMLR*, 2013.

[33] J T Springenberg and M Riedmiller, "Improving deep neural networks with probabilistic maxout units," *CoRR*, 2013.

[34] A Quattoni and A Torralba, "Recognizing indoor scenes," in *CVPR*, 2009, pp. 413–420.

[35] J Deng, W Dong, R Socher, L-J Li, K Li, and L Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.

[36] A Mahmood, M Bennamoun, S An, and F Sohel, "Resfeats: Residual network based features for image classification," *CoRR*, 2016.

[37] B Zhou, A Lapedriza, A Khosla, A Oliva, and A Torralba, "Places: A 10 million image database for scene recognition," *PAMI*, 2017.

[38] S Hochreiter and J Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.