A. Model Architecture of VGG network

Figure A. 1 shows the VGG architecture we used in our experiments.

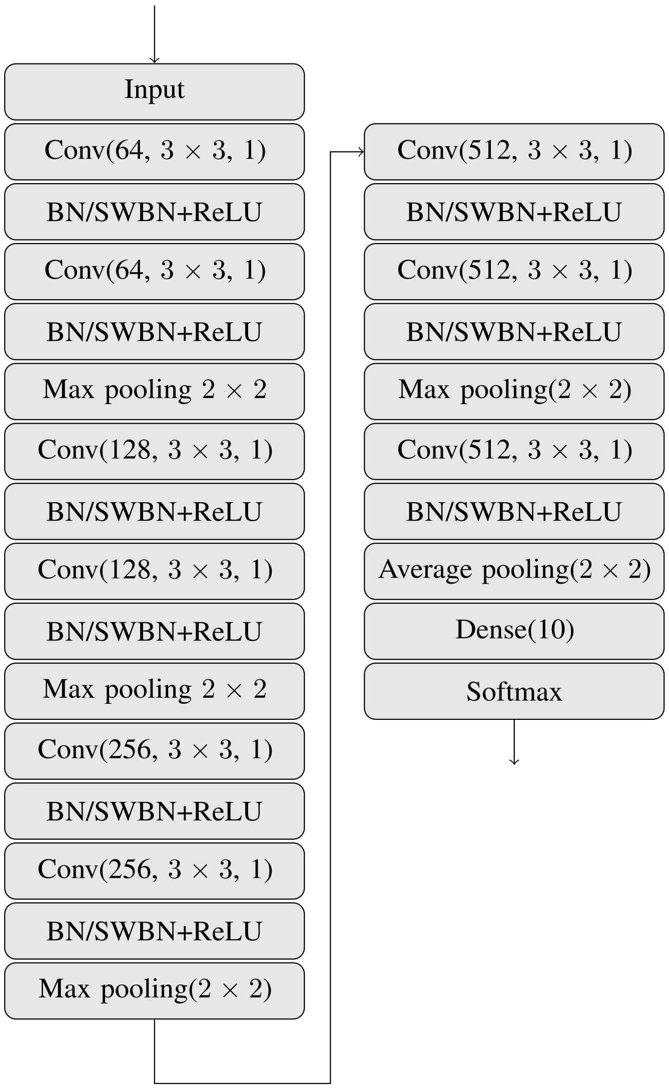

Figure A. 1: VGG architecture. Conv(c, $k \times k$, s) represents a convolutional layer with c channels, kernel of size $k \times k$ and stride s. Dense(h) represents a fully connected layer with h neurons. BN/SWBN represents a normalization layer. In our experiments, we used BN for baseline models and SWBN for our proposed architectures. ReLU represent a rectified linear activation function. Max (Average) pooling ($d \times d$) represents max (average) pooling layer with $d \times d$ kernel.

B. More Details on Whitening Criteria

B.1 Derivation of the update rule for C_{KL}

Given two d-dimensional multivariate Gaussian distributions $p_i \sim \mathcal{N}(\vec{\mu}_i, \Sigma_i)$ and $p_j \sim \mathcal{N}(\vec{\mu}_j, \Sigma_j)$, the KL divergence between them can be simplified as:

$$\begin{aligned} D_{KL} &= \tfrac{1}{2}(\log \tfrac{\det(\Sigma_j)}{\det(\Sigma_i)} + (\vec{\mu}_i - \vec{\mu}_j)^T \Sigma_j^{-1} (\vec{\mu}_i - \vec{\mu}_j) \\ &\qquad + tr(\Sigma_j^{-1}\Sigma_i) - d) \end{aligned}$$

We want to compute the KL divergence between the data distribution and the standard Gaussian distribution. Let p_i stand for the data distribution and p_j for the standard Gaussian distribution with $\vec{\mu}_j = \vec{0}$ and $\Sigma_j = I$. Thus the KL divergence becomes:

$$D_{KL} = \frac{1}{2}(\vec{\mu}_i^T \vec{\mu}_i + tr(\Sigma_i) - \log\det(\Sigma_i) - d)$$

Since the SWBN algorithm finds the whitening matrix W on standardized random vector $\vec{x} \in \mathbb{R}^d$, where $\vec{\mu}_i = E[\vec{x}] = \vec{0}$ and $\Sigma_i = WE[\vec{x}\vec{x}^T]W^T = W\Sigma_{\vec{x}}W^T$, we eventually get the C_{KL} criterion:

$$C_{\text{KL}} = \frac{1}{2}(tr(W\Sigma_{\vec{x}}W^T) - \log\det(W\Sigma_{\vec{x}}W^T) - d)$$

It is shown in [4] that by iteratively updating W using the update rule in Eq. (3), W is optimized to become a whitening matrix. Unlike the classical gradient approach that involves actual gradients, the update rule is derived by the *relative gradient* approach [4]. The reason why we do not use the actual gradient of C_{KL} to update W is due to its high computational cost. The gradient of C_{KL} with respect to W is derived as follows: [1]

$$\begin{aligned} \frac{\partial C_{KL}}{\partial W} &= \frac{1}{2}(\frac{\partial}{\partial W} tr(W\Sigma_{\vec{x}}W^T) - \frac{\partial}{\partial W} \ln\,\det(W\Sigma_{\vec{x}}W^T)) \\ &= \frac{1}{2}(\frac{\partial}{\partial W} tr(W\Sigma_{\vec{x}}W^T) - \frac{\partial}{\partial W}(\ln\,\det(W) \\ &\quad + \ln\,\det(\Sigma_{\vec{x}}) + \ln\,\det(W^T))) \\ &= \frac{1}{2}(2W\Sigma_{\vec{x}} - 2(W^{-1})^T) \\ &= W\Sigma_{\vec{x}} - (W^{-1})^T \end{aligned}$$

As we can see, this gradient formula involves matrix inversion, which is usually computationally expensive and numerically unstable. Therefore, we apply the relative gradient approach, which is introduced in [4], to avoid the matrix inversion. Below we show how using the relative gradient approach yields a more computationally efficient version of the update formula.

Consider a scalar-valued function of a matrix $f : \mathbb{R}^{d\times d} \mapsto \mathbb{R}$, and we want to minimize the function f. In the classical gradient approach, at each step, we find an additive perturbation Γ s.t. $f(W + \Gamma) - f(W) < 0$ and $\Gamma = -\eta\frac{\partial f(W)}{\partial W}$ with a small enough positive number η. In the relative gradient approach, at each step, we aim to find a multiplicative perturbation Γ s.t. $f((I + \Gamma)W) - f(W) < 0$. Gresele et al. [6] has used the following expansion for finding the

[1]For the matrix gradient formulas in steps 1 and 2, please refer to [10].

relative gradient:

$$\begin{aligned} f((I+\Gamma)W) &= f(W) + \left\langle \frac{\partial f(W)}{\partial W}, \Gamma W \right\rangle + o(||W||) \\ &= f(W) + \left\langle \frac{\partial f(W)}{\partial W} W^T, \Gamma \right\rangle + o(||W||) \end{aligned}$$

where $\langle \cdot, \cdot \rangle$ stands for the inner product of matrices. The direction of the steepest decent can be achieved by setting $\Gamma = -\eta \frac{\partial f(W)}{\partial W} W^T$ with a small enough positive number η. Therefore, the additive term to update W is $\Gamma W = -\eta \frac{\partial f(W)}{\partial W} W^T W$, with the relative gradient $\frac{\partial f(W)}{\partial W} W^T W$.

Thus, using the relative gradient, a more computationally efficient version of the update rule of C_{KL} is:

$$\begin{aligned} \frac{\partial C_{\text{KL}}}{\partial W}|_{relative} &= (W\Sigma_{\vec{x}} - (W^{-1})^T)W^T W \\ &= (W\Sigma_{\vec{x}} W^T - I)W \end{aligned}$$

B.2 Derivation of the update rule for C_{Fro}

The update formula for W using C_{Fro} can be directly derived by computing the gradient matrix: [2]

$$\begin{aligned} \frac{\partial C_{Fro}}{\partial W} &= \frac{\partial}{\partial W}\frac{1}{2}||I - W\Sigma_{\vec{x}}W^T||_{Fro} \\ &= \frac{\partial}{\partial W}\frac{1}{2}[tr((I - W\Sigma_{\vec{x}}W^T)^T(I - W\Sigma_{\vec{x}}W^T)]^{\frac{1}{2}} \\ &= \frac{1}{4||I - W\Sigma_{\vec{x}}W^T||_{Fro}}(-2\frac{\partial}{\partial W}tr(W\Sigma_{\vec{x}}W^T) + \\ &\quad \frac{\partial}{\partial W}tr(W\Sigma_{\vec{x}}W^T W\Sigma_{\vec{x}}W^T)) \\ &= \frac{1}{4||I - W\Sigma_{\vec{x}}W^T||_{Fro}}(4W\Sigma_{\vec{x}}W^T W\Sigma_{\vec{x}} - 4W\Sigma_{\vec{x}}) \\ &= \frac{1}{||I - W\Sigma_{\vec{x}}W^T||_{Fro}}(W\Sigma_{\vec{x}}W^T - I)W\Sigma_{\vec{x}} \end{aligned}$$

B.3 Properties of C_{KL} and C_{Fro}

To show the properties of C_{KL}, we first prove the following lemma.

Lemma 1.1. *The set of symmetric positive-definite matrices with bounded eigenvalues is convex. That is to say, let* $\mathbf{V}$ *be a set of all* $d \times d$ *real matrices s.t.* $\forall W \in \mathbf{V}$*:*
(i) $W^T = W$*;*
(ii) $\forall \vec{x} \in \mathbf{R}^d$, $\vec{x}^T W \vec{x} > 0$*;*
(iii) For eigenvalues λ_i *of* W*, there exist* $m, M, 0 < m < M$ *that satisfy* $m \leq \lambda_1(W) \cdots \leq \lambda_d(W) \leq M$*, where* $\lambda_i(W)$ *stands for the* i*-th eigenvalue of* W*.*
The set $\mathbf{V}$ *of matrices is convex.*

[2]For the matrix gradient formulas in steps 3 and 4, please refer to [10].

Proof. For Condition (*i*), let $A, B \in \mathbf{V}$ and a real number $\alpha \in [0,1]$. Define $C = \alpha A + (1-\alpha)B$. It is obvious that $C^T = \alpha A^T + (1-\alpha)B^T = \alpha A + (1-\alpha)B = C$.

For Condition (*ii*), given any non-zero vector $\vec{x}$, we have

$$\vec{x}^T C \vec{x} = \alpha \vec{x}^T A \vec{x} + (1-\alpha)\vec{x}^T B \vec{x}$$

If $\alpha = 0$ or 1, then $\vec{x}^T C \vec{x}$ is equal to either $\vec{x}^T A \vec{x}$ or $\vec{x}^T B \vec{x}$, which must be positive. Otherwise, if $\alpha \in (0,1)$, both $\vec{x}^T A \vec{x} > 0$ and $\vec{x}^T B \vec{x} > 0$, which makes $\vec{x}^T C \vec{x} > 0$.

For Condition (*iii*), we need to prove the upper bound and lower bound of eigenvalues do not change for affine combination of two matrices in $\mathbf{V}$. Let $\lambda(X)$ be the set of eigenvalues of a matrix X. For any matrices A and B in $\mathbf{V}$ and a real number $\alpha \in [0,1]$,

$$\begin{aligned} &\forall e \in \lambda(A), e \in [m,M] \wedge \forall e \in \lambda(B), e \in [m,M] \\ &\Rightarrow \lambda(\alpha(A - mI)) \subseteq [0, \alpha(M-m)] \\ &\wedge \lambda((1-\alpha)B - mI) \subseteq [0, (1-\alpha)(M-m)] \\ &\Rightarrow \alpha(A - mI) \text{ is PSD } \wedge (1-\alpha)(B - mI) \text{ is PSD} \\ &\Rightarrow \alpha(A - mI) + (1-\alpha)(B - mI) \text{ is PSD} \\ &\Rightarrow (\alpha A + (1-\alpha)B) - mI \text{ is PSD} \\ &\Rightarrow \lambda((\alpha A + (1-\alpha)B) - mI) \subseteq [0, \infty) \\ &\Rightarrow \lambda((\alpha A + (1-\alpha)B) \subseteq [m, \infty) \end{aligned}$$

Similarly, we have:

$$\begin{aligned} &\forall e \in \lambda(A), e \in [m,M] \wedge \forall e \in \lambda(B), e \in [m,M] \\ &\Rightarrow \lambda(\alpha(A - MI)) \subseteq [\alpha(m-M), 0] \\ &\wedge \lambda((1-\alpha)(B - MI)) \subseteq [(1-\alpha)(m-M), 0] \\ &\Rightarrow \alpha(A - MI) \text{ is NSD } \wedge (1-\alpha)(B - MI) \text{ is NSD} \\ &\Rightarrow \alpha(A - MI) + (1-\alpha)(B - MI) \text{ is NSD} \\ &\Rightarrow (\alpha A + (1-\alpha)B) - MI \text{ is NSD} \\ &\Rightarrow \lambda(\alpha A + (1-\alpha)B - MI) \subseteq (-\infty, 0] \\ &\Rightarrow \lambda(\alpha A + (1-\alpha)B) \subseteq (-\infty, M] \end{aligned}$$

where PSD and NSD stand for "Positive Semi-Definite" and "Negative Semi-Definite", respectively. Therefore, $\lambda(\alpha A + (1-\alpha)B) \subseteq [m, M]$. □

In addition, we need the following lemma from [11].

Lemma 1.2. *Let* A *be a convex open set and let* $f : A \to \mathbb{R}$ *be a convex function. Then, f is* ρ*-Lipschitz over A if and only if for all* $w \in A$ *and* $v \in \partial f(w)$*, we have that* $||v|| \leq \rho$*. [11]*

Now we are ready to prove the following lemma of C_{KL}'s properties.

Lemma 1.3. *Let* $\Sigma \in \mathbf{R}^{d \times d}$ *be a real positive semi-definite (PSD) symmetric constant matrix. For the whitening criterion*

$$C_{KL} = \frac{1}{2}(tr(W\Sigma W^T) - \ln\det(W\Sigma W^T) - d)$$

defined over $\mathbf{V}$*, the followings hold: (i)* C_{KL} *is convex. (ii)* C_{KL} *is* ρ*-Lipschitz.*

Proof. For any $W \in A$, C_{KL} can be simplified as:

$$C_{\text{KL}} = \tfrac{1}{2}\underbrace{tr(W\Sigma W^T)}_{\text{convex}} + \underbrace{(-\ln\det(W))}_{\text{convex}} - \underbrace{(\frac{1}{2}\ln\det(\Sigma) + \frac{d}{2})}_{\text{constant}}$$

Since Σ is a symmetric PSD matrix, the convexity of the first term is proved in [3]. Because W is a positive definite matrix, the second term in C_{KL} is also convex as shown in [3]. Thus, C_{KL} is convex.

Since W is positive definite (PD) and Σ is PSD, all eigenvalues of these matrices are non-negative. Let λ_M^Σ be the largest eigenvalue of Σ. There exists positive constants $K_1 > 0$, $K_2 > 0$ s.t. $\|W^{-1}\| \leq \frac{K_1}{m}$, $\|W\| \leq K_1 M$, and $\|\Sigma\| \leq K_2\lambda_M^\Sigma$. Recall $\frac{\partial C_{\text{KL}}}{\partial W} = W\Sigma - (W^{-1})^T$, with matrix norm inequalities, we have

$$\left\|\tfrac{\partial C_{\text{KL}}}{\partial W}\right\| = \left\|W\Sigma - (W^{-1})^T\right\| \leq \|W\|\,\|\Sigma\| + \left\|W^{-1}\right\| \leq K_1K_2M\lambda_M^\Sigma + \tfrac{K_1}{m}$$

Let $\rho = K_1K_2M\lambda_M^\Sigma + \frac{K_1}{m}$. Because $\left\|\frac{\partial C_{\text{KL}}}{\partial W}\right\| \leq \rho$ and C_{KL} is convex, with Lemma 1.2, C_{KL} is ρ-Lipschitz. □

Because C_{KL} has these properties over the set $\mathbf{V}$ defined in Lemma 1.1, one might expect C_{KL} to work well as a whitening criterion in a stochastic optimization setting. However, there is one theoretical issue to ensure that C_{KL} preserves these properties. That is how to make the whitening matrix W have bounded eigenvalues, namely, satisfy Condition (*iii*) of Lemma 1.1. Even though we have never observed the maximum eigenvalue of W in SWBN to blow up in practice, our solution to guarantee that the eigenvalues never explode is to constrain the norm of the whitening matrix W. One simple solution is to set a threshold $C > 0$, and compute $tr(W)$ at each step during training. Every time $tr(W) > C$ is observed, then W is replaced by $\frac{C}{tr(W)}W$.

C_{KL} is optimized by the relative gradient approach in SWBN. This approach has been widely used in independent component analysis, as discussed in [13]. Lately, it was introduced to the field of unsupervised learning [6]. Relative gradient descent is a type of gradient descent algorithm on Riemannian manifolds [1]. The convergence of the stochastic relative gradient descent is proved and evaluated in [1] [2].

Unfortunately, C_{Fro} does not have similar properties as C_{KL}. Clearly, C_{Fro} is a non-convex function. For example, in a one dimensional case, with $\Sigma_{\vec{x}} = 1$ and W as a scalar variable w, $C_{Fro} = \sqrt{(1-w^2)^2} = |1-w^2|$, which has two global minima at $w = 1$ and $w = -1$. Even if w is constrained to be positive, namely, w belongs to one dimensional $\mathbf{V}$, $\frac{d^2}{dw^2}C_{\text{Fro}} = -2 < 0$ when $w \in (0,1)$, indicating non-convexity. Similar deduction can be made for the squared version of C_{Fro}, which is $||I - W\Sigma_{\vec{x}}W^T||_{Fro}^2$.

C. Derivation of Back-propagation Algorithm

Let $X \in \mathbb{R}^{d\times n}$ and $\hat{X} \in \mathbb{R}^{d\times n}$ be the input and output of an SWBN layer of a DNN, respectively. Given the loss function L, the gradient matrix $\frac{\partial L}{\partial \hat{X}}$ of L, and $\vec{\mu}$, $\vec{v}$, X^S, X^W from Algorithm 1, we compute the gradient matrix $\frac{\partial L}{\partial X}$ by the chain rule. Without the loss of generality, we derive the formula of the partial derivative $\frac{\partial L}{\partial \hat{X}_{kl}}$ as:

$$\begin{aligned}
\frac{\partial L}{\partial X_{kl}} &= \Sigma_{i=1}^d \frac{\partial L}{\partial \hat{X}_{il}}\frac{\partial \hat{X}_{il}}{\partial X_{kl}} \\
\frac{\partial \hat{X}_{il}}{\partial X_{kl}} &= \frac{\partial}{\partial X_{kl}}(\gamma_i X_{il}^W + \beta_i) = \gamma_i \frac{\partial X_{il}^W}{\partial X_{kl}} \\
\frac{\partial X_{il}^W}{\partial X_{kl}} &= \frac{\partial}{\partial X_{kl}}\Sigma_{t=1}^d W_{it}X_{tl}^S = W_{ik}\frac{\partial X_{kl}^S}{\partial X_{kl}} \\
\frac{\partial X_{kl}^S}{\partial X_{kl}} &= \frac{\partial}{\partial X_{kl}}\frac{X_{kl} - \mu_k(X_{kl})}{\sqrt{v_k(X_{kl}) + \epsilon}} \\
&= \frac{1}{\sqrt{v_k + \epsilon}} + \frac{\partial X_{kl}^S}{\partial v_k}\frac{\partial v_k}{\partial X_{kl}} + \frac{\partial X_{kl}^S}{\partial \mu_k}\frac{\partial \mu_k}{\partial X_{kl}} \\
\frac{\partial X_{kl}^S}{\partial v_k} &= -\frac{1}{2}(X_{kl} - \mu_k)(v_k + \epsilon)^{-\frac{2}{3}} \\
\frac{\partial X_{kl}^S}{\partial \mu_k} &= -\frac{1}{\sqrt{v_k + \epsilon}} \\
\frac{\partial v_k}{\partial X_{kl}} &= \frac{2}{n}(X_{kl} - \mu_k) \\
\frac{\partial \mu_k}{\partial X_{kl}} &= 1/n
\end{aligned}$$

D. Experimental Setup

D.1 ILSVRC-2012

For the ImageNet experiments, we used Stochastic Gradient Decent (SGD) with a batch size of 256 on 8 GPUs. The initial learning rate, momentum, and weight decay are set to 0.1, 0.9 and 10^{-4}, respectively. We follow the same learning rate schedule as described in [7]. We found that replacing all Batch Normalization (BN) layers in the ResNeXt models with Stochastic Whitening Batch Normalization (SWBN) layers slightly decrease the model's test accuracy. We conjecture that all-channel feature whitening probably negatively affects the aggregated transformation [15] in the model where the generalization improvement of ResNeXt originates. To fix this issue, we replace the BN layers that do not have a direct influence on the aggregated transformations, i.e. the first BN layer after the first convolutional layer, as well as the last BN layers in all the residual blocks.

D.2 Few-shot Classification

For the few-shot classification experiments, we re-implement the Cross Attention Network (CAN) [8], and use a Pytorch meta learning library called Torchmeta [5] for the Matching Network (MN) [14] and the Prototypical Network (PN) [12]. For all the whitening layers, due to the small sizes of the backbone networks, we replace shifting parameters $\vec{\gamma}$, which consist of only diagonal elements of Γ, with dense matrices Γ as the task parameters, so the backbone networks using the whitening layers have the equivalent capacity as the networks using BN layers. For experiments of CAN, we follow the experimental configurations as described in [8]. For experiments of PN and MN, we select Adam [9] as the optimization algorithm. Each of the experiments is run for 300 epochs with an initial learning rate equal to 0.001. The learning rate is halved at every 30 epochs. Each training epoch consists of 100 episodes. To get accurate test accuracies, we evaluate all the trained models on 2000 test episodes generated from the test (a.k.a. query) datasets. For each test episode, 30 test samples for each class are randomly sampled from the test dataset.

References

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009. 3

[2] S. Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013. 3

[3] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 3

[4] J.-F. Cardoso and B. H. Laheld. Equivariant adaptive source separation. *IEEE Transactions on signal processing*, 44(12):3017–3030, 1996. 1

[5] T. Deleu, T. Würfl, M. Samiei, J. P. Cohen, and Y. Bengio. Torchmeta: A Meta-Learning library for PyTorch, 2019. Available at: https://github.com/tristandeleu/pytorch-meta. 3

[6] L. Gresele, G. Fissore, A. Javaloy, B. Schölkopf, and A. Hyvärinen. Relative gradient optimization of the jacobian term in unsupervised deep learning. In *NeurIPS 2020 Thirty-fourth Conference on Neural Information Processing Systems*, 2020. 1, 3

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[8] R. Hou, H. Chang, M. Bingpeng, S. Shan, and X. Chen. Cross attention network for few-shot classification. In *Advances in neural information processing systems*, pages 4003–4014, 2019. 3, 4

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[10] K. Petersen and M. Pedersen. *The matrix cookbook*. Technical University of Denmark, Nov 2012. 1, 2

[11] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. 2

[12] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017. 3

[13] S. Squartini, F. Piazza, and A. Shawker. New riemannian metrics for improvement of convergence speed in ica based learning algorithms. In *2005 IEEE International Symposium on Circuits and Systems*, pages 3603–3606. IEEE, 2005. 3

[14] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016. 3

[15] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 3