Deep Learning under Privileged Information Using Heteroscedastic Dropout Supplementary Material

John Lambert*, Ozan Sener*, Silvio Savarese
Department of Computer Science
Stanford University
johnwl,osener,ssilvio@stanford.edu

Abstract

In this supplementary material, we present the proofs for Propositions 1&2 of the main paper and we provide a full derivation of Equation (7) of the main paper, as well as missing implementation details and missing experimental analyses on accuracy vs. percentage of privileged information for image classification.

1. Proof of Proposition 1

The proof of proposition 1 is available at [10] as Example 4. However, we include here a simpler proof for the sake of completeness.

Proof. We will start with

$$\begin{aligned}
&\left| E_{x,y\sim p_{\mathcal{Z}}}[l(y,h(x;w)] - \frac{1}{n}\sum_{i\in[n]} l(y_i,h(x_i;w)\right| \\
&\overset{(a)}{\leq} \left| \sum_{j\in[K]} E[l(y,h(x;w)|(x,y)\in C_j]\mu_j \right. \\
&\qquad \left. - \sum_{j\in[K]} E[l(y,h(x;w)|(x,y)\in C_j]\frac{|n_j|}{n}\right| \\
&\quad + \left| \sum_{j\in[K]} E[l(y,h(x;w)|(x,y)\in C_j]\frac{|n_j|}{n} \right. \\
&\qquad \left. -\frac{1}{n}\sum_{i\in[n]} l(y_i,h(x_i;w)\right| \\
&\overset{(b)}{\leq} \left| \sum_{j\in[K]} E[l(y,h(x;w)|(x,y)\in C_j](\mu_j - \frac{|n_j|}{n})\right| \\
&\quad + \frac{1}{n}\left| \sum_{j\in[K]}\sum_{i\in n_j} E[l(y,h(x;w)|(x,y)\in C_j] - l(y_i,h(x_i;w))\right| \\
&\overset{(c)}{\leq} \left| \sum_{j\in[K]} E[l(y,h(x;w)|z\in C_j](\mu_j - \frac{|n_j|}{n})\right| + \lambda^l\epsilon
\end{aligned}$$

In (a), we use the fact that the space has an ϵ-cover; and denote the cover as $\{C_j\}_{j\in[K]}$ such that each C_j has diameter at most ϵ. We further define an auxiliary variable $\mu_j = p((x,y)\in C_j)$ and $n_j = \sum_i \mathbb{1}[(x_i,y_i)\in C_j]$ and used the triangle inequality. In (b), we use $i\in n_j$ to represent $(x_i,y_i)\in C_j$. Finally, in (c) we use the fact that each ball has diameter at most ϵ and the loss function is λ^l-Lipschitz.

We can bound $E[l(x,y)|z\in C_j]$ with a maximum loss L and use the Breteganolle-Huber-Carol inequality (*cf* Proposition A6.6 of [9]) in order to bound $\sum_j \mu_j - \frac{|n_j|}{n}$.

Combining all, we observe that with probability at least

$1-\delta$,

$$\left| E_{x,y\sim p_{\mathcal{Z}}}[l(y,h(x;w))] - \frac{1}{n}\sum_{i\in[n]} l(y_i,h(x_i;w)) \right| \leq \lambda^l \epsilon + L\sqrt{\frac{2K\log 2 + 2\log(1/\delta)}{n}}$$ □

2. Proof of Proposition 2

The proof of Proposition 2 will closely follow the proof of Proposition 4 and Lemma 5 in [3]. Our main technical tool will be controlling the variance in Bernstein-type bounds to obtain an upper bound which has rate $\mathcal{O}(\frac{1}{n})$. Consider the output of a CNN, given an image as z, with abuse of notation (we used z to represent the representation layer, however for the sake of consistency with [3] we denote z as the output here). Every activation in the neuron can be written as a sum over the paths between the input layer and the representation as $z_i = \sum_p \alpha_p x_p$, where x_p is the input neuron connected to the path and α_p is the weight of the path. One interesting property is the fact that this weight is simply the multiplication of all weights over the path w_p with a binary value. When only max-pooling and ReLU non-linearities are used, that binary value is 1 if all activations are on and 0 if at least one of them is off. This is due to the fact that max-pooling and ReLU either multiply the input with a value of 1 or 0. We call this binary variable $\sigma(x,w)$. Hence, each entry is;

$$z_i = \sum_p x_p \sigma_p(x,w) w_p \tag{1}$$

We can note $\bar{z} = [x_{\bar{0}}\sigma_p(x,w), \cdots, x_{\bar{P}}\sigma_p(x,w)]$ as a vector with dimension equal to the number of paths. Next, we can explicitly compute the generalization bound over l_2 loss as;

$$\begin{aligned} R(A_s) &= E[l(y,h(x;w))] - \frac{1}{n}\sum_i l(y_i,h(x_i;w) \\ &= E[\|h(x;w)-y\|_2^2] - \frac{1}{n}\sum_i \|h(x_i;w)-y_i\|_2^2 \\ &= \sum_c \left(w_c^\intercal \left[E[zz^\intercal] - \frac{1}{n}\sum_i z_i z_i^\intercal \right] w_c \right) \\ &+ 2\sum_c \left(\left[\frac{1}{m}\sum_i y_{i,k} z_i^\intercal - E[y_k z^\intercal] \right] w_c \right) \\ &+ E[y^\intercal y] - \frac{1}{n}\sum_i y_i^\intercal y_i \end{aligned} \tag{2}$$

We will separately bound each term in the following subsections. We first need to prove a useful lemma we will use in the following proofs.

Lemma 1. ***Matrix Bernstein inequality with variance control (corollary to Theorem 1.4 in [8]).*** *Consider a finite sequence $\{M_i\}$ of independent, self-adjoint matrices with dimension d. Assume that each random matrix satisfies $E[M_i] = 0$ and $\lambda_{max}(M_i) \leq R$ almost surely. Let $\gamma^2 = \|\sum_i E[M_i^2]\|_2$. Then, for any $\delta > 0$, if $t \leq \gamma$; with probability at least $1-\delta$,*

$$\lambda_{max}\left(\sum_i M_i\right) \leq \left(\frac{3\gamma + R}{6}\right)\log\frac{d}{\delta}$$

Proof. Theorem 1.4 by Tropp [8] states that for all $t \geq 0$,

$$P\left(\lambda_{max}\left(\sum_i M_i\right) \geq t\right) \leq d\exp\left(\frac{-t^2/2}{\gamma^2 + Rt/3}\right)$$

By using the assumption that $t \leq \gamma$,

$$d\exp\left(\frac{-t^2/2}{\gamma^2 + Rt/3}\right) \leq d\exp\left(\frac{-t/2}{\gamma + R/3}\right)$$

and substituting $\delta = d\exp\left(\frac{-t/2}{\gamma+R/3}\right)$, the result is implied. □

Bounding $z^\intercal z$ term: This will follow directly from the Matrix form of the Bernstein inequality, which is stated as Lemma 1. By using ξ, M_z and P as defined in the main text, Lemma 1 shows that with probability at least $1-\delta$,

$$\lambda_{\max}\left(\left[E[zz^\intercal] - \frac{1}{n}\sum_i z_i z_i^\intercal\right]\right) \leq (\frac{2\xi}{n} + \frac{2M_z}{3n})\log\frac{P}{\delta} \tag{3}$$

By using the definition of the Matrix norm and the Cauchy-Schwarz Inequality, one can show that;

$$\begin{aligned} &\sum_c \left(w_c^\intercal \left[E[zz^\intercal] - \frac{1}{n}\sum_i z_i z_i^\intercal \right] w_c \right) \\ &\leq C\left(\max_c \|w_c\|_2^2\right)(\frac{2\xi}{n} + \frac{2M_z}{3n})\log\frac{P}{\delta} \\ &\leq \frac{2CM_w(3\xi + M_z)}{3n}\log\frac{P}{\delta} \end{aligned} \tag{4}$$

Bounding $z^\intercal y$ term: We need to bound $\sum_c \left(\left[\frac{1}{n}\sum_i y_{i,c} z_i^\intercal - E[y_c z^\intercal]\right] w_c\right)$. In order to bound this term, we will first use the fact that y is a 1-hop vector and the fact that $y_c z^\intercal = z^\intercal$ if $y_c = 1$, and 0 otherwise. Hence,

$$E[y_c z^\intercal] = E[E[y_c z^\intercal | y_c = 1]] = E[z^\intercal | y_c = 1]\mu_c$$

where $\mu_c = p(y_c = 1)$. Using this fact, we can state;

$$\begin{aligned}
&2\sum_c \left(\left[\frac{1}{n} \sum_i y_{i,k} z_i^\intercal - E[y_k z^\intercal] \right] w_c \right) \\
&= 2\sum_c \left(\left[\frac{1}{n} \sum_i y_{i,k} z_i^\intercal - E[z^\intercal | y_c = 1]\mu_c \right] w_c \right) \\
&\overset{(a)}{=} 2\sum_c \frac{|n_c|}{n} \left(\frac{1}{|n_c|} \sum_{i \in n_c} z_i^\intercal - E[z^\intercal | y_c = 1] \right) w_c \\
&\quad + 2\sum_c E[z^\intercal w_c | y_c = 1] \left(\frac{|n_c|}{n} - \mu_c \right)
\end{aligned} \tag{5}$$

In (a), we noted the training examples from class c as n_c. We can use the Bernstein inequality, which states that

$$\begin{aligned}
&p\left(\left[\frac{1}{|n_c|} \sum_{i \in n_c} z_i^\intercal w_c - E[z^\intercal w_c | y_c = 1] \right] > t \right) \\
&\leq \exp\left(-\frac{n_c t^2}{2\xi^2 + \frac{2t}{3}}\right)
\end{aligned} \tag{6}$$

where $\xi^2 = \frac{1}{n_c} \sum_i Var\{z_i\}$. Using the assumption that $Var\{z_i\} \leq \delta$, we can state that with probability at least $1 - \delta$,

$$\begin{aligned}
&2\sum_c \left(\left[\frac{1}{n} \sum_i y_{i,k} z_i^\intercal - E[y_k z^\intercal] \right] w_c \right) \\
&\leq \frac{2C(\xi + 1)}{3n} \log \frac{1}{\delta} + 2C\epsilon_y
\end{aligned} \tag{7}$$

Here ϵ_y is a term which bounds the variance of the class label distribution, which is defined in the next section.

Bounding $y^\intercal y$ term: This term is both independent of the learning algorithm and the weights learned and can be simply made to vanish to zero if the number of samples per class directly follows the population densities. Hence, we do not include a specific rate for this quantity and simply denote it with ϵ_y and assume that it goes to 0 with a rate better or equivalent to a linear rate. See the main text for a detailed explanation as to why we choose to not include ϵ_y in the analysis.

After bounding each term in (2), we can now state the proof for Proposition 2.

Proof. By using the decomposition in (2) and the bounds in (4,7), we can state that

$$\begin{aligned}
&R(A_s) \\
&\leq \frac{2C\left((\xi + 1)\log\frac{1}{\delta} + M_w\left(3\xi + M_z\right)\log\frac{P}{\delta}\right)}{3n} + (2C + 1)\,\epsilon_y
\end{aligned}$$

□

3. Derivation of (7, Main Paper)

In equation (7) of the main paper, we stated that

$$KL(p_w(z|x, x^\star)||p_w(z)) \sim \|\log h^\star(x^\star; w^\star)\| \tag{8}$$

In this section, we formally derive this claim using the log-Uniform assumption. In order to compute $KL(p_w(z|x, x^\star)||p_w(z))$, we need to choose a prior distribution for z. As discussed in depth in [1], the use of ReLU activations empirically suggests that a good choice for this prior would be the log-uniform distribution. Hence, we consider the log-Uniform prior. We first use the definition of the KL-divergence as;

$$\begin{aligned}
&KL(p_w(z|x, x^\star)||p_w(z)) \\
&= -E_{p_w(z|x,x^\star)}[\log p_w(z)] + E_{p_w(z|x,x^\star)}[\log p_w(z|x, x^\star)]
\end{aligned} \tag{9}$$

Since we know the distribution of $p_w(z|x, x^\star)$ as $\mathcal{N}(h^o(x, w^o), h^\star(x^\star, w^\star))$, and using the assumption that the covariance matrix is diagonal,

$$E_{p_w(z|x,x^\star)}[\log p_w(z|x, x^\star)] = \left\| \frac{1}{2}(1 + \log 2\pi h^\star(x^\star; w^\star)) \right\|_1 \tag{10}$$

If we use the log-uniform prior, the first term in the KL-divergence can be computed as;

$$E_{p_w(z|x,x^\star)}[\log p_w(z)] = E_{p_w(z|x,x^\star)}[c_1 + c_2 z] = c \tag{11}$$

where we use the fact that the logarithm of the pdf of a log-uniform distribution is $c_1 + c_2 z$ with appropriate constants. Furthermore, the norm of $h^0(x, w)$ does not affect the output as it is followed with a soft-max operation, which is invariant up-to a scalar multiplication. Hence, we can safely consider its norm to be a constant c. Using both terms,

$$KL(p_w(z|x, x^\star)||p_w(z)) = \bar{c} \log h^\star(x^\star; w^\star)\| - c \tag{12}$$

with appropriate constants $\bar{c}$ and c. We do not include c in the optimization since an additional constant does not change the result of the optimization and we include $\bar{c}$ in the trade-of parameter β.

4. Additional Results

In this section, we provide two experimental results missing in the paper: first, an analysis of accuracy vs. the amount of $x^\star$ provided for image classification and second, a comparison of our method with baselines for the task of ImageNet image classification with 200K images. We also provide further qualitative analysis of the relationship between variance control and our method.

Accuracy vs. Partial $x^\star$ for Image Classification: In the main text, we already studied the case where only a partial

$x^\star$ is available and showed that even a small percentage of $x^\star$ is enough for multi-modal machine translation experiments. Due to the limited space, we provide the same experiment for the image classification here in Figure 1 and show that as long as a small percentage of the dataset has privileged information, our algorithm is effective.

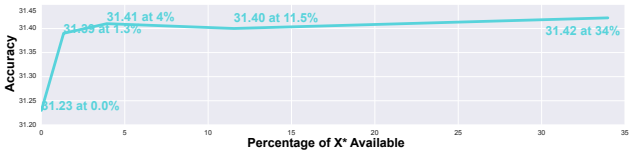

Figure 1. Accuracy vs amount of privileged information ($x^\star$) available for image classification using 75K ImageNet images. We plot top-1, single crop accuracy.

ImageNet with 200K Images In the main paper, we performed the ImageNet image classification experiment with only 75K training images (a mid-sized dataset) and showed that our method learns significantly faster (by reaching higher accuracy) than all baselines. In order to answer the question, *would the result still hold if we had up to 200K images* (a larger-sized dataset) ? We put the results in Table 1 and the results suggest that our method matches the performance of the best baselines in the 200K case. Hence, we can conclude that our method, *i.e.* marginalization, provides no harm in a large-scale dataset regime.

Table 1. We compare our method's performance with several baselines. We train with **200** images per each of the 1000 ImageNet classes, leaving us with 200 $\times 10^3$ images in total. We outperform each model with a significant margin. Since we utilize only 600K (or less) of the 1.28M images from the CLS-LOC ImageNet dataset across all of our experiments, we use a randomly selected subset of the remaining 628K images as a hold-out set for evaluation. Accuracy is given in %, from 0 to 100. Multi-crop accuracy is computed not via individual voting on the correct class by each crop, but rather by taking an `arg max` over classes after summing the `softmax` score vectors of each individual crop.

	Single Crop		Multi-Crop	
Model	top-1	top-5	top-1	top-5
No-$x^\star$ [6]	55.99	79.21	58.60	80.98
GoCNN [12]	50.73	75.39	53.37	77.61
Modal. Hallucination [2]	52.28	76.33	55.66	78.79
Our LUPI	55.20	78.72	58.17	80.90
Gaussian Dropout [7]	55.43	78.77	-	-
MIML-FCN [11]/ResNet-50	56.00	78.83	59.14	81.05
Multi-Task w/ Bbox	**56.32**	**79.45**	**59.29**	**81.48**

5. Additional Qualitative Analysis of the Method

In Figure (2), we visualize the computed variance of the heteroscedastic dropout. Figure (2) supports our hypothesis

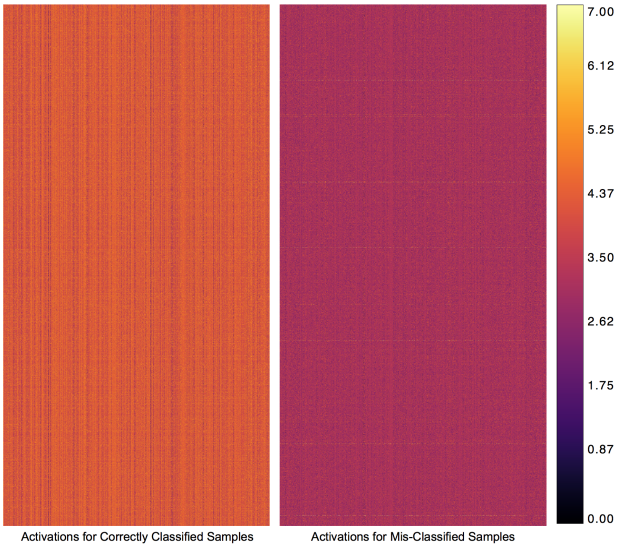

Figure 2. Visualization of the computed variance of our heteroscedastic dropout for 8000 random samples from the validation set that our algorithm mis-classifies, as well as 8000 random samples it correctly classifies. The plot is a heatmap of activations, with dimensions (`num_images`×`num_channels`).

that our algorithm controls the variance since mis-classified examples are expected to have high variance/uncertainty and need to be multiplied with a low value to be controlled. The visualization is fairly uniform, especially for misclassified examples, but we believe the $h^\star$ has interesting information in it which can be further utilized in applications like confidence estimation and is an interesting future work direction.

6. Additional Implementation Details

In this section, we give all of the implementation details of our algorithm, as well as the implementation details of the baselines we used in our experimental study. In order to ensure full reproducibility of all experiments, we share our source code [1]. We found that in all experiments that could converge, from training set sizes of $32K$ up to $600K$ images, an adaptive $10x$ learning rate decay schedule significantly outperforms the traditional 30-epoch fixed $10x$ learning rate decay schedule. We consistently observe performance gains of 510% with the learning rate schedule set adaptively according to whether or not performance on the hold-out validation set has reached a plateau. Unless otherwise noted, we utilize SGD with momentum set to 0.9 for all models, and a learning rate schedule that starts at 1×10^{-2}, as [6] suggests.

Heteroscedastic Dropout Implementation: We set $\lambda = 100$ in all experiments, although we found this was not a

[1] https://github.com/johnwlambert/dlupi-heteroscedastic-dropout

meaningful hyperparameter. We found the training to be prone to convergence in local optima and restarted training if the distribution over class logits was still uniform after 30 epochs. We use a weight decay of 1×10^{-4} in all experiments, ADAM, and a learning rate of 1×10^{-3}, as described in Section 3.2 of the paper. We cropped images to a standard size of 224×224 before feeding them into the network.

We scale the batch size m with respect to the size of the training set. For example, for the 75K model, we use a batch size of 64. For the 200K Model, we use a batchsize of 128. For the 600K model, we utilize curriculum learning and a batch size of 256. We first train the fc layers in the $x^\star$ tower for 8 epochs with ADAM, a batch size of 128, and a learning rate 1×10^{-3}, and then fix the $x^\star$ fc weights and fine-tune the fc layers of the x tower with the ADAM optimizer and a learning rate of 1×10^{-7} and a batch size of 256.

No-$x^\star$: A baseline model without access to any privileged information. We use a batch size of 256.

Gaussian Dropout [7]: We draw noise from $\mathcal{N}\big(\mathbf{1}, diag(\mathbf{1})\big)$ because the authors of [7] state that σ should be set to $\sqrt{\frac{(1-\text{drop prob})}{\text{drop prob}}}$. Empirically, we found that setting $\sigma = 0.1$ performed slightly better. We did not include a regularization loss on the covariance matrices of the random noise. We use SGD with momentum set to 0.9, a learning rate of 1×10^{-2}, and a batch size of 256.

Multi-Task with Bbox: We add one extra head to the VGG network that, just as the classification head, accepts pool5 activations. This regression head produces the center coordinates $(x_{\text{cent}}, y_{\text{cent}})$ and width and height of a bounding box, all normalized to $[0, 1]$. As our loss function, we use a weighted sum of cross entropy loss and $\lambda = 0.1$ times the bounding box regression loss. We use a batch size of 200 instead of 256 because of GPU RAM constraints of ~ 64 GB.

Multi-Task with Mask: In order to predict pixel-wise probabilities between a background and foreground (object) class, we require an auto-encoder network that can preserve spatial information. We experiment with two architectures (DeconvNet) [5] [4]. We chose the DeconvNet architecture for its superior performance, which we attribute to its far greater representation power than DCGAN (the DeconvNet architecture utilizes 15 convolutions instead of the much shallower 5 convolution architecture of the DCGAN generator/discriminator, versus 13 conv. layers in VGG)[4][5] [6]. As our loss function, we use a weighted sum of cross-entropy losses over classes and $\lambda = 0.1$ times the cross entropy loss over masks . We use a batch size of 128 instead of 256 because of GPU RAM constraints of ~ 64 GB.

GoCNN [12] We found that the models could not converge when the suppression loss (computed as the Frobenius norm of the masked activations) is multiplied only by (1/32), as the authors utilize in their work. We found that the model could learn if the suppression loss was multiplied by (1/320) or (1/3200) with ADAM, a learning rate of 1×10^{-3}, and a batch size of 256. We use a black and white (BW) mask for $x^\star$.

Information Dropout [1] As we note in the main paper, we found a VGG-16 network with two Information Dropout layers, each succeeding one of the first two fully connected layers, could only converge with a sigmoid nonlinearity in the fc layers. We keep the ReLU nonlinearity in the convolutional layers. We train with a batch size of 128, set $\beta = 3.0$, set $\alpha_{\text{maximum}} = 0.3$, sample from a log-normal distribution (by exponentiating samples from a normal distribution), and employ an improper log-uniform distribution as our prior, as the authors used for their CIFAR experiments.

MIML-FCN [11]: We compare the use of a VGG-16 or ResNet-50 architecture, with a batch size of 256 and $\lambda = 1 \times 10^{-8}$, which we tuned manually by cross-validation. For the ResNet-50 architecture, we start the learning rate schedule at 1×10^{-1}. We share the convolutional layer parameters across both parameters, and thus find far superior performance when $x^\star$ is provided as an RGB mask, rather than a black and white (BW) mask, because the privileged information is more closely aligned with the input x.

Modality Hallucination [2]: Due to the memory requirements of 3 VGG towers with independent parameters, we chose to share the feature representation in the convolutional layers and to incorporate the hallucination loss between the fc1 activations of the depth and hallucination networks. We use a batch size of 128. For identical reasons as those stated in the previous paragraph, RGB masks are a superior representation for $x^\star$ than BW masks for this model.

References

[1] A. Achille and S. Soatto. Information dropout: Learning optimal representations through noisy computation. *arXiv preprint arXiv:1611.01353*, 2016. 3, 5

[2] J. Hoffman, S. Gupta, and T. Darrell. Learning with side information through modality hallucination. In *Computer Vision and Pattern Recognition*, June 2016. 4, 5

[3] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017. 2

[4] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *IEEE International Conference on Computer Vision*, pages 1520–1528, Dec 2015. 5

[5] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. 5

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4, 5

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 4, 5

[8] J. A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012. 2

[9] A. v. d. Vaart and J. Wellner. Weak convergence and empirical processes with applications to statistics. *Journal of the Royal Statistical Society-Series A Statistics in Society*, 160(3):596–608, 1997. 1

[10] H. Xu and S. Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012. 1

[11] H. Yang, J. Tianyi Zhou, J. Cai, and Y. Soon Ong. Miml-fcn+: Multi-instance multi-label learning via fully convolutional networks with privileged information. In *Computer Vision and Pattern Recognition*, July 2017. 4, 5

[12] J. F. S. Y. Yunpeng Chen, Xiaojie Jin. Training group orthogonal neural networks with privileged information. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 1532–1538, 2017. 4, 5