Learning Visual Variation for Object Recognition

Jatuporn Toy Leksut*, Jiaping Zhao and Laurent Itti

Department of Computer Science, University of Southern California, USA.

ARTICLE INFO

Keywords:
Object Recognition
Multi-Task Learning
Convolutional Neural Network

ABSTRACT

We propose visual variation learning to improve object recognition with convolutional neural networks (CNN). While a typical CNN regards visual variations as nuisances and marginalizes them from the data, we speculate that some variations are informative. We study the impact of visual variation as an auxiliary task, during training only, on classification and similarity embedding problems. To train the network, we introduce the iLab-20M dataset, a large-scale controlled parametric dataset of toy vehicle objects under systematic annotated variations of viewpoint, lighting, focal setting, and background. After training, we strip out the network components related to visual variations, and test classification accuracy on images with no visual variation labels. Our experiments on 1.75 million images from iLab-20M show significant improvement in object recognition accuracy, i.e., AlexNet: 84.49% to 91.15%; ResNet: 86.14% to 90.70%; and DenseNet: 85.56% to 91.55%. Our key contribution is that, at the cost of visual variation annotation during training only, CNN enhanced with visual variation learning is able to focus its attention on distinctive features and learn better object representations, reducing classification error rate of Alexnet by 42%, ResNet by 32%, and DenseNet by 41%, without significant sacrificing of training time and model complexity.

1. Introduction

Object recognition is one of the most fundamental problems in computer vision as it is required in the early stages of processing visual information before a machine can perform other tasks such as searching, tracking, or navigating. To date, convolutional neural networks (CNN) have been shown to be the most effective computational model for object recognition [1]. One of the key strengths of CNN is the ability to learn discriminative features from highly complex image data in an end-to-end manner. Most recent efforts in improving CNNs for object recognition focus on modifying network architecture and have achieved great success at the expense of increasing network complexity, especially its depth and number of parameters [2, 3, 4, 5]. This work takes an orthogonal approach to architectural design by utilizing additional information from training data to enhance the learning capacity of any CNN.

Natural images of an object occur with a wide range of visual variations such as viewpoint, lighting, and background. Visual variations make object recognition more challenging to a machine since they increase the ratio of noise (unwanted information regarding object category) in the image. A typical CNN handles these challenges by requiring a large amount of training data with high variability in order to generalize well over different variations. Some visual variations, however, are informative to humans as they provide cues and context for understanding and interacting with the 3D world. This leads us to question whether a CNN could benefit from observing and recognizing those visual variations instead of disregarding them.

In this work, we introduce a visual variation learning CNN that leverages visual variations in training data to improve representation learning for object recognition. Our networks learn object category as a main task and learn visual variation as an auxiliary task. Two types of visual variation learning are considered: classification and similarity embedding. Visual variation classification is a multi-class classification task, whereas visual variation embedding em-

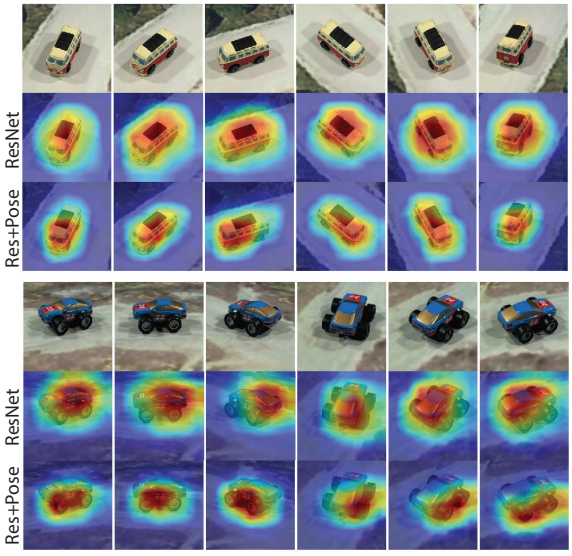


Figure 1: Gradient-weighted class activation mapping (Grad-CAM[6]) on iLab-2M test data. Both object instances are misclassified as car on ResNet without variation learning and correctly classified as van and monster on ResNet with variation learning (ResNet+P). The heatmaps on the second row highlight regions of the image that activate the incorrect class on ResNet and on the third row hightligh regions that activates the correct class on ResNet+P. ResNet+P is more tuned to the shape of an object across poses and focuses on distinctive features of each category (flat front of the van and oversized wheels of the monster truck).

Project website: https://bmobear.github.io/projects/viva/

*Corresponding author

leksut@usc.edu (J.T. Leksut); jiapingz@usc.edu (J. Zhao); itti@usc.edu (L. Itti)

ORCID(s): 0000-0002-4886-1833 (J.T. Leksut)

ploys metric learning to map input data into an embedding space where similarity in terms of visual variation is preserved. The proposed framework is built on top of a simple feed-forward network, AlexNet [1] in our main experiments. We introduce a detachable visual variation learning module that operates during training only and can be removed in test time, so that visual variation only acts as an advisory signal during back-propagation. This feature distinguishes our approach from related works that aim to improve representation learning but require a significant increase in model complexity such as using multiple variation-specific networks [7] or adding generative models [8].

To enable a comprehensive study of visual variation learning, we created the iLab-20M [9] dataset as a valuable addition to object recognition benchmarks. The iLab-20M dataset is a publicly-available large-scale (22M images) controlled parametric dataset of 718 real physical toy vehicles under variations of 1,320 combinations of viewpoint, lighting, focus setting, and backgrounds per object instance. The dataset enables an in-depth analysis of how factors such as viewpoint and lighting impact object recognition. While the ultimate goal of object recognition is to recognize objects in the wild, previous works [9, 10] demonstrate that by using transfer learning, knowledge acquired from iLab-20M can be successfully transferred to natural images in the ImageNet dataset. Similarly, recent works in domain adaptation [11, 12, 13] show that models trained on a controlled image domain can be generalized to perform well on natural image settings.

Our contributions can be summarized as follows:
(1) We propose to improve representation learning in object recognition by explicitly learning visual variation as an auxiliary task in a detachable structure that does not increase model complexity at test time.
(2) We investigate similarity embedding of visual variation as an auxiliary task in addition to a conventional multi-class classification. Our proposed similarity-based networks offer an alternative approach that does not require visual variation categorical labels.
(3) We demonstrate the usefulness of the iLab-20M dataset. Training our networks on iLab-20M, we show that visual variation learning as an auxiliary task improves representation learning for object recognition.

This paper is an extension of the original ideas published in [10] and [14]. We supplement previous works by training and evaluating our models on more comprehensive data sets, introducing visual variation embedding networks, expanding visual variation study to include lighting, and experimenting on data with sparse variations and in-the-wild. Our key finding is that using visual variation as an advisory signal during training significantly improves object recognition models without increasing model complexity.

2. Related Works

Learning visual variation as an auxiliary task falls under the multi-task learning (MTL) paradigm. In the early stages of MTL, Caruana [15] speculated that we can learn a task better if we leverage the information in the training data from related tasks. Over two decades, MTL has been successfully applied in various applications, notably natural language processing[16, 17] and computer vision [18, 19]. MTL in computer vision, combined with the advent of CNN, excels in many vision problems such as face recognition [20], object detection [21], robot grasping [22], pedestrian detection [23], and medical imaging [24].

For object recognition, a relevant and early work [25] proposes feature sharing between objects and their attributes (such as furry, strong, old). The proposed multi-task model learns a shared representation between several visual attributes and object category and achieves better results in object recognition task. While sharing the motivation, our approach does not utilize visual attributes which are often correlated to object appearances. Visual *attribute*, defined as a high-level human-defined semantic description of an object, is meaningful and beneficial to object recognition. In this work, we define visual *variation* to be an observable condition that alters an object appearance when the condition changes, therefore visual variation is not used specifically to describe an object. For example, one can describe a fox with visual *attributes* such as 'furry' or 'four-legged' but not with visual *variations* such as 'blurry', 'half in shadow', or 'viewed from above'. Learning visual variation as an auxiliary task is completely different from learning visual attributes for enhancing object recognition.

The idea to explicitly learn visual variations is closely related to what has been pursued by Hinton et al [26] as they propose to use special neuron units called capsules in a convolutional neural network to encapsulate instantiation parameters such as pose, lighting, and shape deformation. By explicitly representing instantiation parameters with capsules in multiple levels, the proposed architecture aims to preserve spatial relationships of object parts. Our work differs in that we take the MTL approach to learn a shared representation and our visual variation learning module can be attached and detached from the main network as opposed to built-in capsules.

Image understanding is to tease apart the underlying factors of variation instead of disregarding them [27]. This concept inspires a line of research that focuses on image representation disentanglement. Many recent works [28, 29, 8] propose generative models, using variants of auto-encoder and generative adversarial network (GAN), to learn a disentangled representation and show impressive results in manipulating digits, faces, and a small number of 3D rendered objects. Our work does not employ a generative model and instead focuses on improving the power of existing discriminative models without an effect on model complexity.

Another area with active research in representation learning is in the face domain. Recent works have adopted the MTL approach to improve face representation learning in discriminative models [30, 31, 20]. Work by Ranjan et al. [31] develops a multi-purpose model that jointly learns many related tasks including face detection, face alignment, and

Figure 2: The iLab-20M dataset provides toy vehicle images from 15 categories in various viewpoints, lighting conditions, and backgrounds. Cropped images of sample object instances for each category are shown above. Sample images demonstrate rich intra-class variations in shape, size, color, and texture.

visual attribute (gender) recognition. Their prior work [30] is closer to our work in that the tasks include learning visual variation (smile and pose) and identity recognition. Both works propose multi-purpose models that optimize for all tasks and require task-specific sub-networks, which is different from our single-purpose model that considers advisory signals from a detachable visual variation learning at training time only. Work by Yin and Liu [20] is most relevant to ours. Their key idea is to develop pose-invariant face recognition by learning PIE (pose, illumination, and expression) as side tasks. They propose dynamic weighting between tasks and a dynamic routing scheme that directs the network to additionally learn pose-specific identity features (one set of weights per pose) and fuse the results at the end. Our networks, on the contrary, do not require any extra machinery (different weights per pose, routing machinery) at test time.

Besides algorithms and new ideas, one of the major driving forces that advance rapid progress in object recognition is the availability of data. Major object recognition datasets include Caltech-101 [32], CIFAR-10 [33], PASCAL VOC [34], and ImageNet [35]. These datasets provide natural images which are rich in variability but lack of controls in variation parameters. In this work, we present a large-scale controlled dataset that offers variety in visual variations. A previous controlled dataset that is similar to ours is the NORB [36] dataset. NORB offers 5 categories of toy objects under various viewpoints and lighting conditions. The dataset is relatively small, providing 97,200 images from 50 object instances. Our iLab-20M provides 22M images from 15 categories and 718 object instances.

3. The iLab-20M Dataset

A dataset with parameterization of imaging factors is suitable for testing our hypothesis. The iLab-20M dataset provides rich and controlled data on scene and object parameters with high intra-class variability. This enables a comprehensive study of contributing factors on object classification using neural networks. A growing effort in transfer learning and domain adaptation also demonstrates that knowledge learned from controlled datasets is valuable and can be applied to problems in natural scene datasets [11, 12, 13].

iLab-20M [9]: The iLab-20M dataset is a large-scale controlled, parametric dataset of toy vehicle objects under variations of viewpoint, lighting, and background. The dataset is produced by placing a physical object on a turntable and using multiple cameras located on a semicircular arc over the table. Toy vehicle instances are annotated into the following 15 categories: boat, bus, car, equipment, f1car, helicopter, military, monster truck, pickup truck, plane, semi truck, tank, train, UFO, and van. Each category consists of 25 to 160 object instances, and the whole dataset contains 718 object instances. For each object instance, the dataset provides 88 different viewpoints from 11 camera azimuth angles and 8 turntable rotation angles, 5 lighting conditions, 3 camera focus settings, and 14–40 background images (see sample images in 2 and

4. Visual Variation Learning Models

This section describes the architectures of multi-task learning neural networks that jointly learn object category and visual variation. First, we review standard components of a convolutional neural network. Then, we define a visual vari-

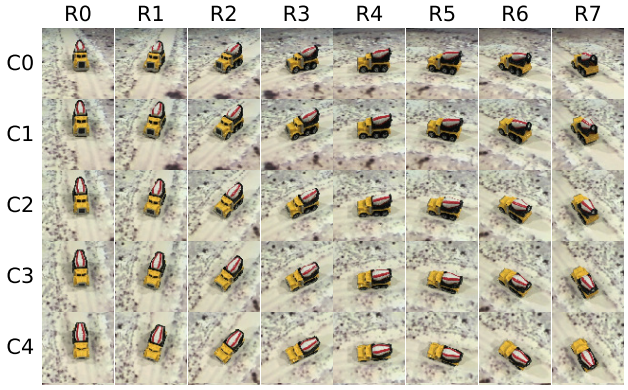


Figure 3: Pose, as a combination of camera viewpoint and object orientation, is one of the visual variations that significantly change the appearance of an object. Shown above are 40 out of 88 available poses from the iLab-20M dataset. Images are produced from 5 cameras varying in azimuth angles and 8 scene rotations.

ation learning module as an add-on component to the main CNN. We propose two types of variation learning module: (1) variation classification and (2) variation embedding.

Convolutional Neural Network (CNN) [37]: A typical convolutional neural network for object recognition consists of a number of computational modules chained together to form a network that is also an acyclic graph. Every computational module is differentiable, which allows the network to adjust its parameters through gradient back-propagation. The most common architecture of CNNs is a chain of conv layers with non-linearity and pooling in between, followed by fully-connected layers (fc) and a loss function. Toward the end, one or more fully-connected layers transform multi-dimensional feature maps into a vector of predicted class scores.

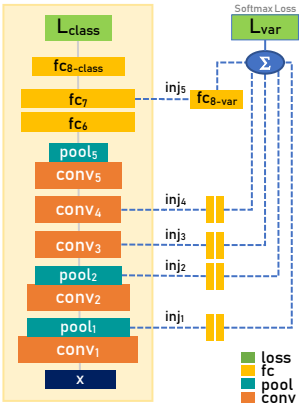
Predicted class scores, represented by an output vector of the final fully-connected layer, become an input to a loss function. For a supervised multi-class classification task, a cross-entropy loss with softmax function is used to compute the distance between predicted class scores $\hat{y}$ and true labels y. A cross-entropy loss with softmax is defined as $-\sum_i y_i log(p_i)$ where $p_i = \frac{e^{\hat{y}_i}}{\sum_{k=1}^{N} e^{\hat{y}_k}}$ is a softmax function that takes predicted class scores $\hat{y}$ and transforms it into a normalized distribution that sums to 1.

In a forward pass, a loss layer computes gradients which are then back-propagated through the network in order to update the parameters. The loss function is important because it defines the objective of the network. It provides feedback on which features are important and how the network should update certain parameters to minimize the loss. In this work, we propose a multi-task object recognition framework that learns visual variation in addition to object category. Our motivation is to use visual variation information as an advisory signal, in the form of an additional loss, to improve parameter updates of the network during training.

4.1. Variation Classification

The variation classification task is to classify a specific visual variation into n_{var} classes. We introduce an injection connection (inj) that connects a hidden layer to a loss function. The injection connection performs data transformation through one or more fully-connected layers to map an input volume of arbitrary size into a vector of a pre-defined size. We use injection connections to add variation classification into a single-task CNN. Note that our use of the term injection refers to injecting variation information during back-propagation and through the loss function at the output layer of the network. That is, variation information is available during training, as an additional supervised teaching signal. This is in contrast to possibly feeding variation information as an additional input to the network, which we do not consider here because it would require that variation information be provided at test time as well.

We build our multi-task model on top of the well-known yet simple feed-forward convolutional neural network AlexNet [1] with minor modifications. Mainly, the size of fc_6 and fc_7 layers changes from 4,096 to 1,024, because the number of classes in our experiment is 15, which is much smaller compared to 1,000 classes from the original AlexNet design.



Layer	Filter Size	Stride, Padding	Output Size
input	–	–	3×224×224
conv_1	11×11	4, 0	96×54×54
pool_1	3×3	2, 0	96×26×26
conv_2	5×5	1, 2	256×26×26
pool_2	3×3	2, 0	256×12×12
conv_3	3×3	1, 1	384×12×12
conv_4	3×3	1, 1	384×12×12
conv_5	3×3	1, 1	256×12×12
pool_5	3×3	2, 0	256×5×5
fc_6	–	–	1024
fc_7	–	–	1024
$\text{fc}_{8\text{-class}}$	–	–	n_{class}

Injection	Input Layer	First FC Size	Second FC Size
inj_1	pool_1	$\text{fc}_{1,1}$: 512	$\text{fc}_{1,2}$: n_{var}
inj_2	pool_2	$\text{fc}_{2,1}$: 512	$\text{fc}_{2,2}$: n_{var}
inj_3	conv_3	$\text{fc}_{3,1}$: 1024	$\text{fc}_{3,2}$: n_{var}
inj_4	conv_4	$\text{fc}_{4,1}$: 1024	$\text{fc}_{4,2}$: n_{var}
inj_5	fc_7	$\text{fc}_{8\text{-var}}$: n_{var}	–

Figure 4: A multi-layer variation-injected CNN built on top of AlexNet [1]. AlexNet is enclosed inside the left box. All additional components on the right are part of the variation classification module. Dotted lines represent injection connections that connect intermediate conv/pool layers to transformation (fc) units. Outputs of transformation units are summed into variation scores.

Our multi-task model with multiple variation injections is represented by a diagram in Figure 4. We add five injection connections, namely inj_1, inj_2, inj_3, inj_4, and inj_5. All except inj_5 connect a 3D input volume to the softmax loss layer, L_{var}, for variation classification. Injection inj_5 shares transformations from layer pool_5 with fc_6 and fc_7. Layer $\text{fc}_{8\text{-var}}$ takes final representation from fc_7 and produces class scores. The final scores for visual variation task are the summation of all injection fc outputs.

During training, the network learns to classify an object into one of n_{class} category classes and one of n_{var} variation classes. The objective function is to minimize the weighted sum of losses:

$$\mathcal{L}_{\text{total}} = (1-\alpha)\mathcal{L}_{\text{class}} + \alpha\mathcal{L}_{\text{var}}$$

where $\alpha \in [0, 1]$. The variation loss layer $\mathcal{L}_{\text{var}}$ takes a weighted sum of the output of the injection transformation units: $\hat{y}^{\text{var}} = \sum_{i=1} \gamma_i t_i$ where $t_i \in \mathbb{R}^{n_{\text{var}}}$ is an output of transformation units at injection i.

Once trained, the network can be deployed without the variation classification module. All AlexNet components can be detached from the injection connections and function as a single-task object recognition network.

4.2. Variation Embedding Learning

In variation classification, we discretize the variation space and assign labels to a certain range. The number of variation classes can be arbitrary, and the final representation does not preserve relationships in terms of variation distance between classes. On the contrary, variation embedding aims to learn a direct mapping from a visual variation space to an embedding space that preserves variation distance. For example,

given three images $X_{0°}$, $X_{15°}$, $X_{60°}$ of an object facing 0°, 15°, 60° away from the camera, we may classify them into three different variation classes and lose their relationship in terms of pose similarity. With variation embedding, however, we can map them into an embedding space where $X_{15°}$ remains closer to $X_{0°}$ than $X_{60°}$.

We introduce visual variation embedding not as a competitor to variation classification but rather an alternative approach that can be applied in different scenarios. For example, consider a video of a turning car where pose gradually changes between frames, instead of classifying pose in every frame, we can assign similar and dissimilar pairs of poses based on consecutive frames. Variation classification requires variation labels and sufficient training data to represent each class, whereas variation embedding allows representation mapping without requiring the specific variation categorical labels. A similarity-based method is suitable for a problem that has a large number of classes, a small number of samples per class, and unseen classes at test time [38].

In this work, we learn a similarity function of visual variation as an auxiliary task. We retain our single-task network AlexNet for category classification and attach a similarity-based network to specifically learn visual variation embedding. Next, we present two widely-used variations of similarity-based network: the Siamese and the triplet.

4.2.1. Siamese Embedding

The Siamese network [39] trains on pairs of images to minimize distances between similar pairs and maximize distances between dissimilar pairs. The network consists of two identical networks with shared parameters. Given an input pair for the same object instance $(x^{id}_{v1}, x^{id}_{v2})$ and its binary label y where $y = 0$ for a similar variation and $y = 1$ for a dissimilar variation, the network optimizes a contrastive loss function [40]:

$$\mathcal{L} = (1-y)\frac{1}{2}D^2 + y\frac{1}{2}max\{0, m-D\}^2$$

where $D = \|G(x^{id}_{v1}) - G(x^{id}_{v2})\|_2$ is the Euclidean distance between the outputs of G, G is a mapping function that maps an input x to an embedding space, and $m > 0$ is a margin that defines the minimum distance between dissimilar pairs. The network is trained to estimate G.

4.2.2. Triplet Embedding

Given a triplet $(x^{id}_{v}, x^{id}_{v+}, x^{id}_{v-})$ in which x^{id}_{v} is an anchor image, x^{id}_{v+} is a positive image of the same instance with similar variation, and x^{id}_{v-} is a negative image of the same instance with dissimilar variation, the network optimizes a triplet loss function [41]:

$$\mathcal{L} = [\|G(x^{id}_{v}) - G(x^{id}_{v+})\|^2_2 - \|G(x^{id}_{v}) - G(x^{id}_{v-})\|^2_2 + m]_+$$

where $[z]_+ = max(z, 0)$, G is a mapping function that maps an input x to an embedding space, and $m > 0$ is a margin value that enforces the minimum difference in distances between the anchor to positive and negative samples. The margin value is set to 1. The auxiliary network is trained to estimate G.

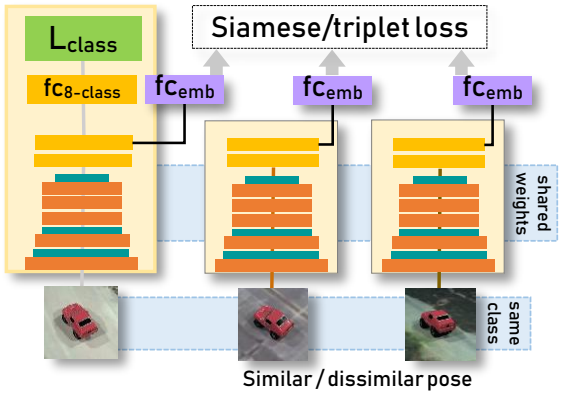


Figure 5: A conceptual diagram of an embedding network. The main AlexNet network takes an anchor input and produces class probability scores. For Siamese setting, the second network in the middle takes a second input of the same object instance but with either similar or dissimilar pose. For triplet setting, the second and the third network takes one similar pose and one dissimilar pose. Both auxiliary networks output an embedding vector representing a point in pose embedding space.

Once trained, the network can be deployed without the auxiliary network. The main AlexNet can fully function as a single-task object recognition network. This is important because the resulting single-task object recognition network has exactly the same structure and number of degrees of freedom (parameters) as the original single-task network, and the only difference is the weight values learned.

5. Experiments and Results

From iLab-80M, we generate our own subsets for experiments conducted in this work. Our focus variation is pose, therefore we select data that vary in pose variation and keep other visual variations constant. With this criterion, we create iLab-2M which contains 30 different poses from a combination of 5 elevations from camera and 6 azimuths from rotation. We sample a total of 1,751,719 images (1.7M) and partition them into 70% training (1.2M), 15% validation (270K), and 15% test (270K) splits. Object instances in training, validation, and test splits are non-overlapping. The iLab-2M dataset is publicly available to download from `http://ilab.usc.edu/ilab2m/iLab-2M.tar.gz`.

Data Processing: The size of an input image is 256×256. We apply random crop of size 224×224 during training and center crop of size 224×224 during validation and test. We do not mirror our data during training due to potential confusion with pose labels. Input data are normalized channel-wise to have zero mean and a standard deviation of 0.5. Data are shuffled every epoch during training.

Training Protocol: We use the Caffe [42] framework to conduct experiments in section 5.1 and PyTorch [43] in all other sections. We train our networks on the training split of iLab-2M and use the validation split to select hyperparameters and

Table 1 & Figure 6: Test Accuracy By Multi-Layer Injection Networks. (Left) s_{0-4} are random seeds for five initial parameters; *s.e.* is standard error; *Err.Red.* is percent error reduction compared to baseline. (Right) Corresponding boxplot displays the distribution of test accuracy from five initial parameters and the ensemble results.

Model	Test Accuracy on Category on iLab-2M							Err. Red.
	s_0	s_1	s_2	s_3	s_4	Mean±s.e.	Ensemble	
I_0	82.60	81.93	81.73	80.63	81.71	81.72±0.32	84.41	–
$I_{[1]}$	81.70	77.78	82.18	81.88	81.48	81.00±0.81	84.69 (+0.28)	1%
$I_{[12]}$	81.95	82.97	82.80	83.51	83.30	82.91±0.27	85.97 (+1.56)	10%
$I_{[123]}$	83.31	85.52	83.83	84.34	84.08	84.22±0.37	87.40 (+2.99)	19%
$I_{[1234]}$	83.21	84.04	84.52	84.29	85.22	84.26±0.33	87.02 (+2.61)	16%
$I_{[12345]}$	*85.19*	84.34	83.59	84.00	*86.16*	84.66±0.46	87.67 (+3.26)	20%
$I_{[2345]}$	84.13	85.28	84.83	84.22	84.09	84.51±0.23	87.23 (+2.83)	18%
$I_{[345]}$	83.66	84.50	83.34	84.75	83.69	83.99±0.27	87.30 (+2.89)	18%
$I_{[45]}$	84.61	84.21	84.10	83.45	84.69	84.21±0.22	87.23 (+2.82)	18%
$I_{[5]}$	84.31	*85.60*	*85.64*	*85.04*	83.63	**84.84**±0.39	**88.10** (+3.69)	23%

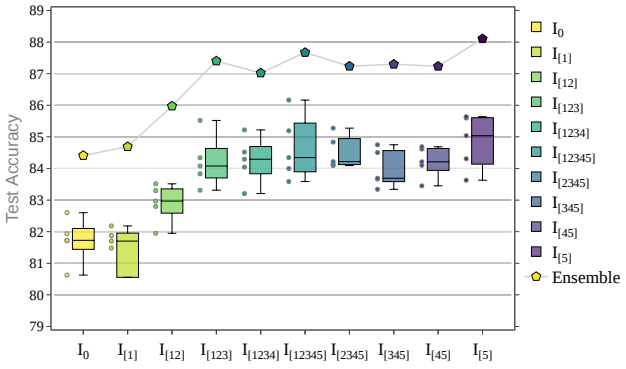
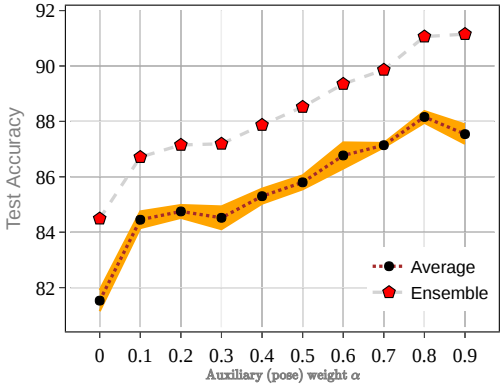


Table 2 & Figure 7: (Left) Test Accuracy By Visual Variation Loss Weight α. (Right) Average and ensemble category classification accuracy of $I_{[5]}$ on different pose weight α. The total loss is $(1-\alpha)\mathcal{L}_{\text{category}} + \alpha\mathcal{L}_{\text{pose}}$. Both $\mathcal{L}_{\text{category}}$ and $\mathcal{L}_{\text{pose}}$ have the same order of magnitude.

α	Test Accuracy on Category on iLab-2M							Err. Red.
	s_0	s_1	s_2	s_3	s_4	Mean±s.e.	Ensemble	
0	81.97	80.11	81.07	82.09	82.43	81.53±0.38	84.49	–
0.1	85.15	83.81	83.51	84.95	84.83	84.45±0.30	86.71 (+2.22)	14%
0.2	85.23	83.92	85.20	84.42	84.96	84.75±0.23	87.15 (+2.66)	17%
0.3	85.73	83.84	85.23	84.57	83.21	84.52±0.41	87.19 (+2.70)	17%
0.4	85.32	86.34	84.89	84.57	85.36	85.30±0.27	87.87 (+3.38)	21%
0.5	86.65	85.27	85.76	85.17	86.15	85.80±0.25	88.52 (+4.03)	25%
0.6	86.78	87.74	85.60	85.61	*88.13*	86.77±0.47	89.35 (+4.86)	31%
0.7	87.24	87.12	87.04	87.43	86.88	87.14±0.08	89.86 (+5.37)	34%
0.8	87.64	*88.57*	*88.82*	*88.00*	87.78	**88.16**±0.21	91.06 (+6.57)	42%
0.9	*89.01*	86.99	87.61	86.87	87.23	87.54±0.35	**91.15** (+6.66)	42%



measure the training progress. We use stochastic gradient descent (SGD) with 0.9 momentum to update network parameters with a batchsize of 16 and initial learning rate of 0.001. To ensure validity of the experiments, every network produces five copies that are trained on five different initial weights and random seeds. We stop the training when the classification accuracy on the validation set plateaus. We then test our trained networks on the test split of iLab-2M.

Training Techniques: We investigate the effectiveness of different training techniques on our baseline AlexNet model and conclude that the best combination of techniques for training our classification networks on iLab-2M is to perform batch normalization [44] before ReLU, initialize weights with a standard Gaussian distribution, and use PReLU [45] activation function.

5.1. Visual Variation Classification

Here we discuss the implementation of the variation classification module. We present ablation studies on multi-layer injections and loss weight assignment.

5.1.1. Multi-Layer Injection Architectures

As described in Section 4.1 and depicted in Figure 4, the auxiliary module contains five injection connections, each connects a convolutional or pooling layer to a softmax loss layer for visual variation. We perform an experiment on various injection combinations to study the impact of multi-task regularization on different layers. We simply switch on and off each one of the five connections and observe changes in test accuracy of the main category classification task. We investigate the 10 following settings: $I_{[1]}$, $I_{[12]}$, $I_{[123]}$, $I_{[1234]}$, $I_{[12345]}$, $I_{[2345]}$, $I_{[345]}$, $I_{[45]}$, $I_{[5]}$, and I_0. The naming convention represents the injection connections that are active. We distribute the weights equally for the total loss (weighted sum of $\mathcal{L}_{\text{class}}$ and $\mathcal{L}_{\text{var}}$) and the final visual variation scores (weighted sum of all inj).

Results: We report object recognition results for five initial weights, the average±the standard error, and the ensemble results in Table 1 (Figure 6). We compute the ensemble results by summing up all five class probability scores.

On average, all injection architectures except $I_{[1]}$ outperform the baseline I_0. The leading architectures are $I_{[5]}$ and $I_{[12345]}$. The ensemble results of $I_{[5]}$ achieves 88.10% in test accuracy, 3.69% higher than the baseline. $I_{[12345]}$ achieves comparable ensemble results of 87.67% in test accuracy, 3.26% higher than the baseline. Looking at the boxplot in Figure 6, we observe a trend that bottom-layer injections provide less benefit compared to top-layer injections. CNN's bottom layers typically encode abstract representations such as edges or corners, and the top layers encode more of high-level features that are specific to problem target domains. The results on multi-layer injections suggest that regularizing with multiple losses towards the bottom layers is less effective and could be destructive to the main task as demonstrated by $I_{[1]}$. In terms of training overhead, $I_{[5]}$ performs much better than $I_{[12345]}$. On $I_{[5]}$, the number of parameters only increases by 0.31% (31k from 10M) and the change in training time is minimal and in practice negligible. On the contrary, $I_{[12345]}$ has 17.5 times more parameters (175M from 10M), and the training time overhead is 43% higher compared to I_0.

5.1.2. *Loss Weight Assignment*

In multi-task learning, a model's performance is sensitive to relative loss weights between tasks [46]. In this experiment, we explore how weight shifting between the main and the auxiliary task impacts the model's performance. We define the total loss to be the weighted sum of individual losses: $\mathcal{L}_{\text{total}} = (1-\alpha)\mathcal{L}_{\text{category}} + \alpha\mathcal{L}_{\text{pose}}$. Both $\mathcal{L}_{\text{category}}$ and $\mathcal{L}_{\text{pose}}$ are cross-entropy loss with softmax, and both losses share the same order of magnitude throughout training. We previously set $\alpha = 0.5$ in all other experiments unless specified otherwise. For this experiment, we shift the value of α from 0 to 1 in an increment of 0.1 and observe test accuracy changes on network $I_{[5]}$. We report test results in Table 2 (Figure 7).

Figure 7 shows the impact of loss weight balancing on the category classification task. Increasing weight of auxiliary task increases the performance of the main task, which is counter-intuitive but supports our hypothesis that the main task benefits from the auxiliary task by allowing higher degree of regularization. We found that the impact of pose loss continues to grow beyond the equal share of weight ($\alpha > 0.5$). In fact, the best performance on average and from the ensemble occurs when allowing more than 80% of the total loss from the auxiliary task. The best ensemble result from $\alpha = 0.9$ achieves 91.15% test accuracy, 6.66% better than the baseline from $\alpha = 0$.

5.2. Variation Embedding Learning

In this section, we describe our experiments on visual variation embedding learning. Unless specified otherwise, most of the experimental setups are identical to our variation classification module explained in Section 5.1. We set the size of the embedding layer fc_{emb} to 32, the margin values for the Siamese embedding to 0.5, and the margin values for the triplet embedding to 1.

Figure 8: Sample data randomly selected by the pair and triplet training protocol. First row images are anchors. Second row images are positive samples (same object instance with similar pose). Third row images are negative samples (same object instance with dissimilar pose).

5.2.1. *Siamese Embedding*

As illustrated in Figure 5, our Siamese embedding network takes a pair of images of the same object instance but with different poses as an input. We use all the images from the iLab-2M training split. The process of pair generation goes as follows. For each image in the training split, we randomly select two more images from the training set that are from the same object instance but one with similar pose and the other with dissimilar pose. Our criterion on similarity is based on whether two images are immediate neighbors in terms of rotation label. For a dissimilar image, the distance between rotation and camera angle must be greater than three neighbors away (refer to Figure 3). Following this pair selection protocol, we generate 2.4M pairs of images, which means in one epoch the network processes a total of 4.8M forward passes.

5.2.2. *Triplet Embedding*

We want to compare the triplet embedding network with the Siamese embedding network, so we control the training images. We directly adopt the generated pairs from the Siamese training data and group them into triplets. The first image from the training set is designated an anchor. The second image with a similar pose becomes a positive sample. The third image with a dissimilar pose becomes a negative sample. This way, our triplet network receives the same training information as the Siamese network. Following this triplet selection protocol, we generate 1.2M triplets of images, which means in one epoch the network processes a total of 3.6M forward passes. Because the parameters are shared among all networks, the number of parameters only increases by 0.33% (33k from 10M). The training time overhead per one forward pass is minimal and negligible. The total training time for a single epoch, however, increases in proportion to the number of forward passes.

Results: We report test accuracy on the object recognition task using the iLab-2M test split in Table 3. During test, we detach the variation embedding components and keep the main AlexNet with the learned weights. The Siamese and

Table 3
Test Accuracy By Visual Variation Embedding Networks

	Test Accuracy on iLab-2M							Err. Red.
	s_0	s_1	s_2	s_3	s_4	Mean±s.e.	Ensemble	
I_0	79.42	81.84	79.35	80.49	80.97	80.41±0.42	82.53	–
Siamese	*82.32*	81.99	82.48	*81.81*	81.47	82.01±0.16	83.43 (+0.90)	5%
Triplet	82.03	*82.25*	*82.75*	81.67	*82.36*	**82.21**±0.16	**83.61** (+1.08)	6%

triplet embedding networks show comparable performance, and they both consistently outperform the baseline. On average, Siamese and triplet embedding networks improve test accuracy by 1.6% and 1.8% respectively. Ensemble results show similar improvement by 0.90% and 1.08%.

We restrict the training samples from pair generation to be relatively small (only adding one positive and one negative samples per original image), because we want to study immediate effects on similarity embedding training. There are several techniques in training Siamese and triplet networks that we have not explored such as hard-negative data mining and online pair selection. Incorporating these techniques could further improve the results.

5.3. Extension to ResNet and DenseNet

We show that our approach can be applied to other CNN architectures besides AlexNet. We build multi-task models for ResNet-50[4] and DenseNet-121[5] by attaching the pose classification module to the last fully-connected layer ($I_{[5]}$ equivalent). We use pre-trained weights from ImageNet as the starting point and finetune each network on iLab-2M. Images are normalized channel-wise to have mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. We use SGD optimizer on a batch size of 16 with an initial learning rate of 0.001 and a momentum of 0.9. The α is set to 0.8.

Table 4
Classification Test Accuracy of ResNet and DenseNet with visual variation learning. $+P$ is pose learning.

	Test Accuracy on iLab-2M							Err. Red.
	s_0	s_1	s_2	s_3	s_4	Mean±s.e.	Ensemble	
Res	86.11	82.72	86.45	83.60	82.87	84.35±0.72	86.14	–
Res+P	*89.54*	*88.16*	*87.96*	*88.09*	*88.80*	**88.51**±0.26	**90.70** (+4.56)	32%
Den	84.99	82.69	85.68	84.22	82.84	84.08±0.52	85.56	–
Den+P	*89.03*	*88.93*	*87.79*	*89.59*	*88.71*	**88.81**±0.26	**91.55** (+5.99)	41%

Results: We report test accuracy in Table 4. The multi-task networks with visual variation learning consistently outperform the baseline on both ResNet and DenseNet. On average (and ensemble), ResNet and DenseNet achieve 4.16% (4.56%) and 4.73% (5.99%) higher in test accuracy. The error rates are reduced by 32% for ResNet and 41% for DenseNet, while the number of parameters only increases by 0.26% (61k from 23M) on ResNet and 0.44% (30k from 7M) on DenseNet. The training time overhead is minimal and negligible on both networks.

5.4. Extension to Lighting Variation

In this section, we show that our method can be applied to other visual variations presented in the iLab-20M dataset and can also be extended to include more than one visual variations. We create an extension of the iLab-2M dataset to include all five lighting conditions shown in Figure 9 in addition to 30 poses. This dataset, namely iLab-2M-Light, is more challenging than iLab-2M as the network has to learn to disentangle composite variations where each variation imposes effects on one another. We sample a total of 1,999,743 images and partition them into 70% training (1.36M), 15% validation (316K), and 15% test (316K) splits. The extended iLab-2M-Light dataset is publicly available to download from `http://ilab.usc.edu/ilab2m/iLab-2M-Light.tar.gz`.

We evaluate our method on iLab-2M-Light under the following settings.

1. I_0: Single-task learning as a baseline
2. $I_{[5]}$+L: Light classification as an auxiliary task
   $\mathcal{L}_{\text{total}} = 0.5\mathcal{L}_{\text{category}} + 0.5\mathcal{L}_{\text{light}}$
3. $I_{[5]}$+P: Pose classification as an auxiliary task
   $\mathcal{L}_{\text{total}} = 0.5\mathcal{L}_{\text{category}} + 0.5\mathcal{L}_{\text{pose}}$
4. $I_{[5]}$+LP: Two auxiliary tasks (light and pose)
   $\mathcal{L}_{\text{total}} = 0.33\mathcal{L}_{\text{category}} + 0.33\mathcal{L}_{\text{light}} + 0.33\mathcal{L}_{\text{pose}}$

Table 5
Test Accuracy of Multiple-Variation Learning

	Test Accuracy on iLab-2M-Light							Err. Red.
	s_0	s_1	s_2	s_3	s_4	Mean±s.e.	Ensemble	
I_0	78.25	79.91	79.62	77.88	79.16	78.96±0.39	80.54	–
$I_{[5]}$+L	82.56	82.11	82.93	81.33	82.50	82.29±0.27	83.90 (+3.36)	17%
$I_{[5]}$+P	85.25	84.72	85.60	84.69	85.17	85.08±0.17	86.75 (+6.22)	31%
$I_{[5]}$+LP	*85.35*	*85.44*	*85.92*	*86.22*	*85.54*	**85.69**±0.16	**87.33** (+6.79)	34%

Results: Training and testing our models on iLab-2M-Light, we report the results in Table 5. All visual variation classification models $I_{[5]}$+L, $I_{[5]}$+P, and $I_{[5]}$+LP consistently outperform the baseline I_0, on average (and ensemble) achieving 3.32% (3.36%), 6.12% (6.21%), and 6.73% (6.79%) higher in test accuracy. For networks with one auxiliary visual variation, $I_{[5]}$+P performs better than $I_{[5]}$+L. When learning two visual variations, $I_{[5]}$+LP achieves the best results.

5.5. Learning with Sparse Variations

Our prior experiments assume that visual variation data are available and equally distributed. In this section, we evaluate visual variation learning on sparse variation data which

Figure 9: Lighting variation in iLab-2M-Light.

is more representative of real world data. We create iLab-sparse, a subset of iLab-2M with 5 different poses, and build four subsets where each object instance comes in a few variations but not all. We experiment on four settings where the number of variations per instance, n_{VPI}, ranges from one to four. The number of training samples are 40k, 80k, 120k, and 160k respectively. We train I_0 and $I_{[5]}$ on each training split from scratch and then test on the same test split from iLab-sparse.

Table 6
Test Accuracy of Different Number of Variations Per Instance (n_{VPI})

n_{VPI}		Test Accuracy on iLab-sparse							Err. Red.
		s_0	s_1	s_2	s_3	s_4	Mean±s.e.	Ensemble	
1	I_0	72.64	74.20	*75.33*	*76.35*	*75.24*	**74.75**±0.56	**77.77**	–
	$I_{[5]}$	*74.19*	*74.83*	74.35	73.87	73.29	74.11±0.23	77.40 (-0.37)	-2%
2	I_0	78.47	78.29	80.30	*79.75*	77.84	78.93±0.42	80.97	–
	$I_{[5]}$	*80.86*	*81.98*	*80.50*	79.54	*79.81*	**80.54**±0.39	**82.86** (+1.89)	9%
3	I_0	82.04	82.32	79.32	80.98	81.06	81.14±0.47	83.51	–
	$I_{[5]}$	*84.00*	*82.97*	*82.67*	*82.66*	*83.16*	**83.09**±0.22	**84.46** (+0.95)	5%
4	I_0	82.16	83.06	83.64	81.60	81.57	82.41±0.37	84.07	–
	$I_{[5]}$	*84.85*	*85.47*	*84.27*	*83.42*	*84.41*	**84.48**±0.30	**85.36** (+1.29)	8%

Results: We report the results in Table 6. Models with visual variation learning $I_{[5]}$ outperforms the baseline I_0 in all settings except when only a single variation is available per object instance (n_{VPI}=1). Both networks learn better as n_{VPI} increases. This suggests that our approach might not work well if the number of variations per instance is too sparse. To improve the visual variation learning would come at the cost of collecting more visual variation data.

5.6. Knowledge Transfer to ImageNet3D+

We finetune our iLab-2M pre-trained models (I_0^{pre} and $I_{[5]}^{\text{pre}}$) on ImageNet3D+ images from PASCAL3D+ dataset[47] to demonstrate that the knowledge gained from an auxiliary task in $I_{[5]}^{\text{pre}}$ can transfer to in-the-wild data. We also investigate knowledge transfer coupled with visual variation learning ($I_{[5]}^{\text{pre}}$+P). ImageNet3D+ provides 3D annotations (object elevation, azimuth, and distance to the camera) for original ImageNet data. This allows us to collect visual variation labels. In this experiment, we classify object pose into 8 classes based on azimuth angle, and we use all 12 object categories for the main task.

Table 7
Classification Accuracy on ImageNet3D+

	Test Accuracy on ImageNet3D+							Err. Red.
	s_0	s_1	s_2	s_3	s_4	Mean±s.e.	Ensemble	
I_0^{pre}	86.78	86.50	87.12	86.97	86.77	86.83±0.09	89.02	–
$I_{[5]}^{\text{pre}}$	*88.05*	*87.88*	*88.36*	*88.04*	*88.27*	**88.12**±0.08	**90.02** (+1.00)	9%
$I_{[5]}^{\text{pre}}$+P	88.03	87.25	88.18	87.77	87.72	87.79±0.14	89.67 (+0.65)	5%

Results: We report the results in Table 7. The iLab-2M pre-trained network enhanced by visual variation learning $I_{[5]}^{\text{pre}}$ transfers better features to ImageNet3D+ (reducing classification error rate by 9%), but the performance degrades when coupled with visual variation learning during finetuning. One possible reason is that ImageNet3D+ has a single variation per instance and the network struggles in this case as shown earlier in Section 5.5.

6. Discussion and Conclusion

We propose to improve object recognition with CNN by incorporating visual variation learning as an auxiliary task. Our finding is that CNNs enhanced with visual variation learning learn better representations. Our experiments on AlexNet, ResNet, and DenseNet show that visual variation learning reduces object classification error rate by 42%, 32%, and 41%. We visualize network activation maps (see Figure 1) and speculate that visual variation learning helps the network focus its attention on distinctive features and thus perform better in recognition tasks.

Our approach is lightweight and effective. The training overhead is minimal when attaching the auxiliary module to the top fully-connected layers, and the number of parameters increases by less than 0.5% on these networks during training, and by 0% at test time. However, the main trade-off is that it requires visual variation labels, which might be scarce in real world data. Our experiment with sparse variation shows that the network struggles on one variation per sample but performs better when more variations are available. One option to mitigate the lack of data is to perform data augmentation by rendering more training samples using 3D models as demonstrated in [48]. Another option is to train a separate network to estimate visual variation labels.

Another possible approach, given that we have visual variation information at hand, would be to use pose, for example, as an input to the network during training. In such case, we would also need to input pose information at test time as well. Our approach, on the other hand, considers using pose information during training only. By design, the visual variation learning module can be detached from the main object classification network at test time.

For visual variation classification, we experiment with multi-layer injection architectures and find that regularizing top hidden layers with the auxiliary task loss helps improve the network's generalization. We also study the impact of weight loss balancing and find that the main task benefits more from multi-task learning as the the ad-hoc weighting of the auxiliary loss increases.

For visual variation similarity embedding, our experiment shows that pose similarity embedding improves the network performance on the main task. We conjecture that more rigorous training techniques such as hard-negative mining and online pair selection would further improve the results. One may ask: why not generate pairs and triplets based on object category in addition to pose, for example, a triplet of an anchor, a negative sample of same category but different pose, and a positive sample of same pose but different category? We consider such variation and decide that it is not desirable to train a network to estimate the same func-

tion with conflicting objectives since the triplet loss will keep positive samples of same pose but different categories close to each other while the main classification network will attempt the opposite.

Finally, we show the usefulness of the iLab-20M dataset that provides rich information, especially in terms of parameter control, which allows us to disentangle visual variations and perform a comprehensive study on object recognition. We extend visual variation learning to include pose and lighting and show that both variations contribute to the network's performance improvements.

Acknowledgment

This work was supported by the National Science Foundation (grant number CCF-1317433), C-BRIC (one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA), and the Intel and CISCO Corporations. The authors affirm that the views expressed herein are solely their own, and do not represent the views of the United States government or any agency thereof. Computation for the work described in this paper was also supported by the University of Southern California's Center for High-Performance Computing (hpc.usc.edu).

References

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

[2] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, CoRR abs/1409.1556 (2014).

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[4] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[5] G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

[6] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proc. IEEE International Conference on Computer Vision, 2017, pp. 618–626.

[7] H. Su, S. Maji, E. Kalogerakis, E. G. Learned-Miller, Multi-view convolutional neural networks for 3d shape recognition, in: Proc. IEEE International Conference on Computer Vision, 2015.

[8] X. Chen, X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in: D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems 29, 2016, pp. 2172–2180.

[9] A. Borji, S. Izadi, L. Itti, ilab-20m: A large-scale controlled object dataset to investigate deep learning, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2221–2230.

[10] J. Zhao, C. K. Chang, L. Itti, Learning to recognize objects by retaining other factors of variation, in: IEEE Winter Conference on Applications of Computer Vision, 2017, pp. 560–568.

[11] E. Tzeng, J. Hoffman, T. Darrell, K. Saenko, Simultaneous deep transfer across domains and tasks, in: IEEE International Conference on Computer Vision, IEEE, 2015, pp. 4068–4076.

[12] B. Sun, K. Saenko, Deep coral: Correlation alignment for deep domain adaptation, in: European Conference on Computer Vision, Springer, 2016, pp. 443–450.

[13] T. Gebru, J. Hoffman, L. Fei-Fei, Fine-grained recognition in the wild: A multi-task domain adaptation approach, in: IEEE International Conference on Computer Vision, IEEE, 2017, pp. 1358–1367.

[14] J. Zhao, L. Itti, Improved deep learning of object category using pose information, in: IEEE Winter Conference on Applications of Computer Vision, 2017, pp. 550–559.

[15] R. Caruana, Multitask learning, in: Learning to learn, Springer, 1998, pp. 95–133.

[16] R. Collobert, J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, in: Proc. 25th International Conference on Machine Learning, ACM, 2008, pp. 160–167.

[17] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, Journal of Machine Learning Research 12 (2011) 2493–2537.

[18] I. Misra, A. Shrivastava, A. Gupta, M. Hebert, Cross-stitch networks for multi-task learning, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3994–4003.

[19] I. Kokkinos, Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2017, pp. 5454–5463.

[20] X. Yin, X. Liu, Multi-task convolutional neural network for pose-invariant face recognition, IEEE Transactions on Image Processing (2017).

[21] R. Girshick, Fast r-cnn, in: Proc. IEEE International Conference on Computer Vision, ICCV '15, IEEE Computer Society, Washington, DC, USA, 2015, pp. 1440–1448.

[22] L. Pinto, A. Gupta, Learning to push by grasping: Using multiple tasks for effective learning, in: IEEE International Conference on Robotics and Automation, IEEE, 2017, pp. 2161–2168.

[23] Y. Tian, P. Luo, X. Wang, X. Tang, Pedestrian detection aided by deep learning semantic tasks, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5079–5087.

[24] S. Chaichulee, M. Villarroel, J. Jorge, C. Arteta, G. Green, K. McCormick, A. Zisserman, L. Tarassenko, Multi-task convolutional neural network for patient detection and skin segmentation in continuous non-contact vital sign monitoring, in: IEEE International Conference on Automatic Face & Gesture Recognition, IEEE, 2017, pp. 266–272.

[25] S. J. Hwang, F. Sha, K. Grauman, Sharing features between objects and their attributes, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2011, pp. 1761–1768.

[26] G. E. Hinton, A. Krizhevsky, S. D. Wang, Transforming auto-encoders, in: International Conference on Artificial Neural Networks, Springer, 2011, pp. 44–51.

[27] Y. Bengio, et al., Learning deep architectures for ai, Foundations and trends® in Machine Learning 2 (2009) 1–127.

[28] S. Reed, K. Sohn, Y. Zhang, H. Lee, Learning to disentangle factors of variation with manifold interaction, in: International Conference on Machine Learning, 2014, pp. 1431–1439.

[29] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, Y. LeCun, Disentangling factors of variation in deep representation using adversarial training, in: D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems 29, 2016, pp. 5040–5048.

[30] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, R. Chellappa, An all-in-one convolutional neural network for face analysis, in: IEEE International Conference on Automatic Face & Gesture Recognition, IEEE, 2017, pp. 17–24.

[31] R. Ranjan, V. M. Patel, R. Chellappa, Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (2019) 121–135.

[32] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: An incremental bayesian approach tested

on 101 object categories, Computer Vision and Image Understanding 106 (2007) 59–70.

[33] A. Krizhevsky, Learning multiple layers of features from tiny images (2009).

[34] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International Journal of Computer Vision 88 (2010) 303–338.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.

[36] Y. LeCun, F. J. Huang, L. Bottou, Learning methods for generic object recognition with invariance to pose and lighting, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, volume 2, IEEE, 2004, pp. II–104.

[37] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.

[38] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, volume 1, IEEE, 2005, pp. 539–546.

[39] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a" siamese" time delay neural network, in: Advances in Neural Information Processing Systems, 1994, pp. 737–744.

[40] R. Hadsell, S. Chopra, Y. LeCun, Dimensionality reduction by learning an invariant mapping, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, volume 2, IEEE, 2006, pp. 1735–1742.

[41] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.

[42] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, arXiv preprint arXiv:1408.5093 (2014).

[43] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, in: NIPS-W, 2017.

[44] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.

[45] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proc. IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.

[46] A. Kendall, Y. Gal, R. Cipolla, Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[47] Y. Xiang, R. Mottaghi, S. Savarese, Beyond pascal: A benchmark for 3d object detection in the wild, in: IEEE Winter Conference on Applications of Computer Vision, 2014.

[48] H. Su, C. R. Qi, Y. Li, L. J. Guibas, Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views, in: Proc. IEEE International Conference on Computer Vision, 2015, pp. 2686–2694.