Neural Dataset Generality

Ragav Venkatesan
Arizona State University
ragav.venkatesan@asu.edu

Vijetha Gattupalli
Arizona State University
jgattupa@asu.edu

Baoxin Li
Arizona State University
baoxin.li@asu.edu

Abstract

Often the filters learned by Convolutional Neural Networks (CNNs) from different datasets appear similar. This is prominent in the first few layers. This similarity of filters is being exploited for the purposes of transfer learning and some studies have been made to analyse such transferability of features. This is also being used as an initialization technique for different tasks in the same dataset or for the same task in similar datasets. Off-the-shelf CNN features have capitalized on this idea to promote their networks as best transferable *and* most general *and are used in a cavalier manner in day-to-day computer vision tasks.*

It is curious that while the filters learned by these CNNs are related to the atomic structures *of the images from which they are learnt, all datasets learn similar looking low-level filters. With the understanding that a dataset that contains many such* atomic structures *learn general filters and are therefore useful to initialize other networks with, we propose a way to analyse and quantify generality among datasets from their accuracies on transferred filters. We applied this metric on several popular character recognition, natural image and a medical image dataset, and arrived at some interesting conclusions. On further experimentation we also discovered that particular classes in a dataset themselves are more general than others.*

1. Introduction

Neural networks, particularly CNNs have broken all records recently in the computer vision research area. The growth of CNNs focused initially on the recognition of characters. Fukushima and LeCun were the initial pioneers. Independently they developed CNN based systems, some of which are still being used widely [7, 16]. Large networks are often trained with large number of data samples to achieve good accuracies [25, 14]. Still, scepticism over CNNs among the modern day computer vision scientists stems from the fact that one does not have a clear understanding of its inner working. Some studies show that a few ($< 1\%$) nodes are all that are actively contributing to classification [4]. They also suggest that large networks often overfit, but since the data is too large over-fitting often works as an advantage [20].

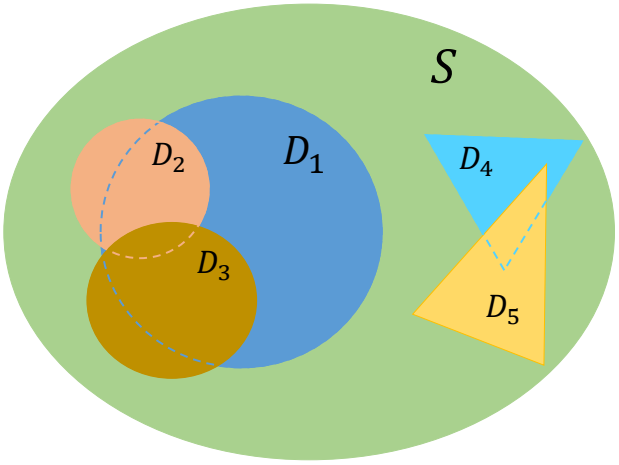

Figure 1. Thought experiment to describe the dataset generality. S is the space of all possible atomic structures, $D_1 - D_5$ are the atomic structures present in respective datasets.

While it is reasonable to expect edge detectors and Gabor-like features in the lower-level filters and more sophisticated concepts at the higher levels, it is not clear as to why these filters adapt themselves in this manner. What is fairly clear though is that different datasets result in different sets of filters that are similar if the datasets are similar. It is only natural to ask, what role does the data itself play in such filters being learnt and how they compare with filters learnt from another dataset. In this paper we take the view that the filters learnt by networks when trained using a particular dataset represent the detectors for some *atomic structure* in the data itself. In which case each layer is a mapping form the previous layer to the next layer that is constructed using combinations of these atomic structures in the first layer in order to minimize a cost.

Let us first define *atomic structures* to be the forms that CNN filters take by virtue of the entropy of the dataset it is learning on, analogous to dictionary atoms. Complex

datasets have more and varied atomic structures. Consider the following thought experiment: Let's assume that all possible atomic structures reside in an universe S. Suppose we have a set of three datasets $D = \{D_1, D_2, D_3\}$ and $D \in S$. Consider the system in figure 1. The figure describes the configuration of the elements of D. One would now recognize that D_1 is a more general dataset with respect to D_2 and D_3. It is so because, while D_1 *contains* most of the atomic structures of D_2 and D_3, the latter do not contain as many atomic structures of D_1. While this analysis is simplified for one layer, in typical CNNs, co-adaptation plays a major role in the learning of these atomic structures. Therefore, generality as defined by the overlap of areas in a layer-wise Venn diagram is impractical to obtain.

In this paper we postulate that, the generalization performances of CNNs on one dataset re-trained on a network initialized by training using another, could be used to derive generality. We call this process of pre-training as *prejudicing*. By prejudicing on the first dataset, we froze and unfroze layers and retrained the networks on the second dataset. By freezing layers we are making a network more obstinate and we call this process obstination[1]. The more the layers are frozen, the more obstinate the feature extractor is, therefore the harder the classifier has to work. If the prejudice was general enough, the classifier shall still generalize fairly well enough. What this means is that if the prejudicing dataset is more general than the re-train dataset, the classifier can generalize better than vice versa.

We developed a generality metric by comparing the gain in performances of networks of various obstination. Using a generality such as the one proposed, it becomes clearer as to what kind of datasets are to be used to prejudice CNNs with during transfer learning. We even discovered that samples with particular labels within a dataset alone are general enough. So, if we begin by prejudicing the network on only those and then moved on to the rest of the labels, we were able to learn the rest of the dataset with considerably less training samples while achieving comparable generalization performances.

Off-the-shelf networks such as VGG, overfeat and various published Caffe model weights are trained on large scale image datasets such as Imagenet or PASCAL [23, 12, 8, 22, 5]. For instance, while these may work on applications such as human pose recognition or vehicle detection, they do not necessarily work on tasks involving medical images. This is because the datasets on which they are trained are not general enough to adapt to the representational requirements of medical images, which is on a manifold unique and disjoint form the manifolds of natural images. This is visualized in D_4 and D_5 from figure 1. Even a large collection of natural images is not general enough to have networks trained that are suitable to medical images. In these cases, the prejudiced network often fails. For instance, on the Colonoscopy dataset discussed later a 22 layer deep overfeat features, trained with a logistic regression performs poorer than a 3 layer deep CNN trained from random initialization, which is in turn outperformed when initialized by a network trained on an endoscopy dataset.

In this article we considered popular offline character recognition datasets and arrived at some interesting analysis and generalities. We also show that within the MNIST dataset, classes $[4, 5, 8]$ are general enough that we could learn the other classes with very few (even just one) samples, when prejudiced with networks trained on $[4, 5, 8]$. We also considered more sophisticated datasets such as Cifar 10 and Caltech 101 against some medical image datasets for colonoscopy video quality [13]. This study led us to two major research insights:

1. If one has very few data to learn from, which other dataset is better to prejudice the network with? The answer is particularly helpful when dealing with medical image datasets where data is very scarce and one can't simply use a network trained on VOC datasets as feature extractors as discussed above.

2. Among the various classes during the training procedure, if we prejudice with a certain *general* set of classes first and then move on to others later, generalization to all classes, even for those with few samples is better. This is particularly significant if the dataset has a lot of samples in certain classes and not as much of others.

The rest of the paper is organized as follows: section 2 discusses related works, section 3 presents the design of our experiments, section 4 shows some results on the core-experiment and section 5 provides concluding remarks.

2. Related work

One related work that this article shares with is the work by Yosinski et al [26]. In that article, the authors considered two tasks A and B that were essentially 500 classes each from the Imagenet dataset [22]. They trained an 8 layer network on one of the tasks (say A). They then initialized a new network carrying over the first n layers from the previous job while randomly initializing the others. This new network was used to retrain task B. Such a network was AnB^+. They experimented by obstination of the carried over layers. Such a network was AnB. They also studied the specificity of each layer and their contributions to the overall performance. They also showed that networks working on similar tasks had a high memorability and that co-adaptation of layers increased the generalization performance.

[1] Obstinate layer or freezing implies that the weights were not changed during backprop. The layer remains prejudiced.

While this analysis is interesting, it was performed on only one dataset: Imagenet. By design, the networks were forced to learn very general filters, so as to be best transferable. Since the images were all *natural images*, one would expect the layers to be more Gabor-like at earlier layers and have more label specific features at later layers, which was what was observed. Also, the paper analysed the transferability of the feature extractors from the perspective of the networks in terms of their fall in generalization performance. This analysis was not catered to the dataset's perspective, which is that the filters learned are a property of the dataset being trained on. This was not a problem for the authors as their datasets for tasks A and B occupied similar manifolds. This analysis also didn't explore re-training using the same network but rather went with re-initializing so that they could learn new co-adaptations. This is not interesting to the study of generality as we want to observe the effect of filters transferred from one dataset on another. The more *general* a dataset, the more variety of atomic structures it offers to the network to learn. We used this idea to define a generality metric between two datasets. To do so, we cannot follow the techniques used by Yosinski et al.

Another closely related work is the work on *dark knowledge* by Hinton et al, [10]. Here the authors suggest that among the various classes in a dataset, there exists some amount of generalization knowledge that could be transferred. The authors construct a large network that learns all its classes. They then go on to train a smaller network with the same dataset (or with a dataset that is missing some of the classes altogether). While training this smaller network though, instead of using the the hard labels, they also use the softmax output from the large network also for backprop. This creates an effect of the larger network *guiding* the smaller network to not just generalize to the dataset, but also to generalize to unseen classes. This is because, as the argument goes, "the network learns the relationship between the classes" and "all the knowledge is among the relative probabilities or softmaxes that the network is almost certain is wrong" [10].

Although the author retrains an entire network that is randomly initialized using the softmax outputs from a trained network and uses this as prejudice, no information is actually being transferred in terms of actual filters. Ergo, this work, while interesting, also doesn't help in understanding generality of the data itself in a more direct manner. Some of the claims made by this article though were indirectly and independently verified by us through our generality results. The basic claim of their work is that among only a handful of classes, there is enough knowledge to generalize to other classes. Unless there exists some generality between classes, training on particular classes will not have been representational enough for the other classes to learn on. We directly verify this by showing that some classes alone have a high generalization to the rest of the dataset and make a similar conclusion from an entirely independent direction of research.

Figure 3. Samples of some of the datasets that we used in this analysis. From top to bottom: MNIST [17], MNIST-rotated [15], MNIST-random-background [15], MNIST-rotated-background [15], Google street view house numbers [19], Char 74k English [3], Char 74k Kannada [3]. Last two rows, first five from left are CIFAR 10 and the rest are Caltech101 [13, 6]. The bottom row is the colonoscopy dataset.

3. Design of experiments

Consider figure 3. Among the various datasets shown, it is natural to expect any network trained on MNIST to contain simpler filters than MNIST-rotated. This is because, while MNIST-rotated contains many structures from MNIST, due to the rotations, MNIST-rotated will contain additional structures that require the learning of more *complicated* filters. A network trained on MNIST-rotated on its first layers will be expected to additionally have filters for detecting sophisticated oriented edges than for MNIST. This would mean that prejudicing a network with MNIST to then re-train MNIST-rotated is much less helpful than vice versa. A network prejudiced with a general enough dataset is better to be retrained for it generalizes easily. A prejudice must come from a more general dataset if a prejudice transfers positive knowledge as shown in their generaliza-

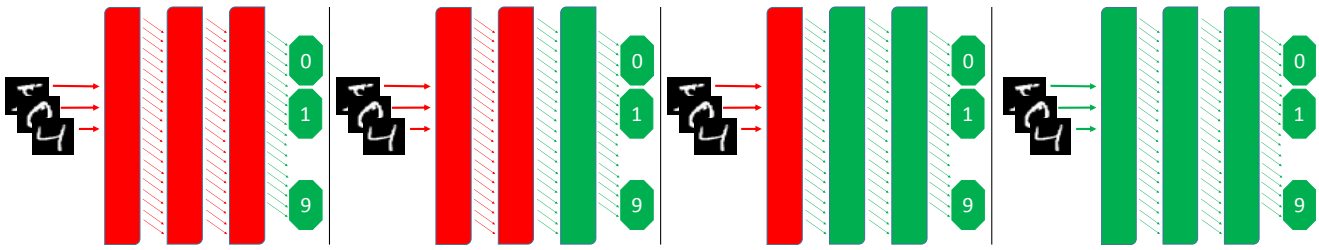

Figure 2. Protocol of obstination: From left to right, all layers frozen, one, two and three layers unfrozen. Green represent unfrozen and red represent frozen. Note that the layers are always unfrozen from the end and that the softmax layer is always unfrozen and randomly initialized. This should be generalized similarly for more than three layers also.

tion performances. We use this simple intuition to argue that MNIST-rotated is a more general dataset with respect to MNIST.

Our basic experiment is conducted between pairs of datasets D_i and D_j. Firstly, we train (prejudice) a randomly initialized network with dataset D_i. We call this network $n(D_i|r)$ or the base network (r implies random initialization). We then proceed to retrain $n(D_i|r)$ as per any of the setup shown in figure 2. $n_k(D_j|D_i)$ would imply that there are k degrees of freedom, or to be precise, k layers of filters that are allowed to learn by dataset Dj that is prejudiced by the filters of $n(D_i|r)$. $n_k(D_j|D_i)$ has $N-k$ obstinate layers that carries the prejudice of dataset D_i, where N is the total number of layers. Note that more degrees of freedom implies that the network is less obstinate to learn. Also note that these layers can be both convolutional or fully connected neural layers. Any idea expressed here can be extended to any type of parametrized layers. In fact while we perform operations such as batch normalizations, we even freeze and unfreeze the α and β of batch norm [11]. Obstination also includes the bias parameters.

Layers learn in two facets. They learn some components that are purely their own and some that are co-adapted from previous layers that are allowed to learn as well. By freezing some layers we are making those layers a fixed functional transformation. Note that the performance gain from $n_k(D_j|D_i)$ and $n_{k+1}(D_j|D_i)$ is not because of just the new layer $k+1$ being allowed to learn, but of the combination of all $k+1$ layers allowed to learn.

Figure 2 shows the setup of our experiments and explains degrees of freedom. These are our obstination protocols. Notice that in all the various setup, the softmax layer remains non-obstinate. In fact the softmax layer is always randomly re-initialized because not all dataset pairs have the same number of labels. Also notice that the unfreezing of layers happen from the rear. We cannot unfreeze a layer that feeds into a frozen layer. This is because, while the unfrozen layer learns a new filter and therefore represents the image on new distributed domains, the latter layer is not adapting to such a transformation. When there are two layers unfrozen, the two layers should be able to co-adapt together and must finally feed into an unfrozen classifier layer.

3.1. Dataset generality

Suppose the generalization performance of $n(D_j|r)$ is $\Psi(D_j|r)$ and the generalization performance of $n_k(D_j|D_i)$ is $\Psi_k(D_j|D_i)$. First order dataset generality or simply dataset generality of D_i with respect to D_j at the layer k is given by,

$$g_k(D_i, D_j) = \frac{\Psi_k(D_j|D_i)}{\Psi(D_j|r)} \tag{1}$$

This indicates the level of performance that is achieved by D_j using $N-k$ layers worth of prejudice from D_i and k layers worth of features from D_i combined with k layers of novel knowledge from D_j together. Note that the generality is calculated for the base dataset as a measure of how the re-train performs with the prejudice of the base dataset. $g_k(D_i, D_j) > g_k(D_i, D_l)$ indicates that at k layers, D_i provides more general features to D_j than to D_l. Conversely, when initialized by $n(D_i|r)$, D_j has an advantage in learning than D_l.

Note that, $g_k(D_i, D_i) \geq 1 \ \ \forall k.\ g_k(D_i, D_j)$ for $i \neq j$ might or might not be greater than 1. If $g_k(D_i, D_j) \geq 1$ for $i \neq j$, it indicates that D_j is at least very similar to D_i (such as the case considered by Yosinski et al.) and at most a perfect generalizer of D_i [26].

3.2. Class generality

D_i and D_j need not be entire datasets but can also be just disjoint class instances of the same dataset that is split in two. These generalities will tell us if particular classes are themselves more general than others. For instance, we divided the MNIST dataset into two parts. The first part contained the classes $[4, 5, 8]$, the rest were contained by the second part[2]. We performed the generality experiments

[2]We chose this combination of classes strategically after trail and error as these are the most general among the classes and exaggerate the effect.

with MNIST$[4, 5, 8]$ as base, which was trained over a random initialization. We re-trained this prejudiced network using the second part with the same experiment design as above. We defined class generality as the generality, of a class or a set of classes, retrained on the prejudice of the other mutually exclusive classes.

We repeated this experiment several times with decreasing number of training samples per-class in the retrain dataset of MNIST $[0, 1, 2, 3, 6, 7, 9]$. All the while, the testing set remained the same size. This implies that the prejudiced network retrains on a much smaller dataset and tests on a much larger dataset. The re-train dataset had 7 classes. We created seven such datasets with $7p$, $p \in [1, 3, 5, 10, 20, 30, 50]$ samples each. We now define subclass generality as the generality of these sub-sampled datasets (in each class we only consider a small random sample), retrained on the base of other mutually exclusive classes (MNIST$[4, 5, 8]$). . Initializing a network that was trained on only a small sub-set of well-chosen classes can significantly improve generalization performance on all classes, even if trained with arbitrarily few samples, even at the extreme case of one-shot learning.

3.3. Datasets Used

We designed these experiments across three board categories of datasets: 1. Character datasets that included MNIST [17], MNIST-rotated [15], MNIST-random-background [15], MNIST-rotated-background [15], Google street view house numbers [19], Char 74k English [3] and Char 74k Kannada [3] 2. Natural image datasets that includes Cifar 10 and Caltech 101 [13, 6] and 3. Natural images against medical images that included in addition to Caltech 101 a Colonoscopy video qualitty dataset. We leave it to the reader to find for themselves details about the datasets from the original articles, but the setup we have used can be found in table 1. Although we chose only a handful of datasets, the intention of this article was only to show that such generality measures could be made. The scope of this article was not to benchmark various publicly available popular datasets. Neither was it to make suggestions specific to types of datasets.

3.4. Network architecture and learning

We used one standard network architecture for all character datasets and experiments, one for Cifar 10 vs. Caltech 101 and another standard for Caltech 101 vs. Colonoscopy. The setup we have used can be found in table 1.

The network architectures, learning rates and other details are provided below. The experiments were conducted on a Macbook Pro Laptop using an Nvidia GT 750M GPU, for character datasets and on an Nvidia Tesla K40 GPU for the others, with cuDNN v3 and Nvidia CUDA v7.

Table 1 shows the train-test-validation splits and the batch sizes used in stochastic gradient descent of all the datasets used. No pre-processing were done on the images themselves except for cropping, resizing, normalizing. The images were all normalized to lie in $[0, 1]$. The character recognition datasets were all of a constant $28X28$ grayscale, the Caltech 101 vs. Cifar 10 experiments were performed ar $32X32$, RGB and the Caltech 101 vs. Colonsoscopy were at $128X128$, RGB. It is to be noted that the aim of the authors was not to set up the networks to achieve state-of-the-art. The authors did although try to achieve satisfactory performances on all base datasets involved before proceeding with the experimentation.

Character Datasets.

Our networks had three convolutional layers with $20, 20$ and 50 kernels respectively. All the filters were 5 X 5 and were all stride 1 convolutions. The first layer didn't have any pooling. The second and the third layer maxpool by 2 subsampled. All the layers used rectified linear units ($ReLU$) activations [18]. The classifier layer was a softmax layer and we didn't use any fully connected layers. We used a dropout of 0.5 only from the last convolutional layer to the softmax layer [24]. We optimized a categorical cross-entropy loss using an rmsprop gradient descent algorithm [2]. For acceleration we used Polyak Momentum that linearly increases in range $[0.5, 1]$ from start to 100 epochs [21]. Unless early terminated, we ran 200 epochs. We also used a constant L_1 and L_2 regularizer co-efficients of 0.0001. Our learning rate was a 0.01 with a multiplicative decay of 0.0998.

CIFAR10 Vs. Caltech101 and Caltech 101 vs Colonoscopy.

For this task, the networks had five convolutional layers with $20, 20$, 50, 50 and 50 kernels respectively. We also had a last fully connected layer of 1800 nodes, which also had a dropout of 0.5. All the filters were 5 X 5 and were all stride 1 convolutions. Only the last layer maxpool by 2 subsampled. All the layers used rectified linear units ($ReLU$) activations [18]. All CNN and MLP layers were also batch normalized [11].The classifier layer was a softmax layer and we didn't use any fully connected layers. We used a dropout of 0.5 only from the last convolutional layer to the softmax layer [24]. We optimized a categorical cross-entropy loss using an rmsprop gradient descent algorithm [2]. For acceleration we used Polyak Momentum that linearly increases in range $[0.5, 0.85]$ from start to 100 epochs [21]. We use a learning rate of 0.001 for the first 150 epochs and then fine tune with a learning rate of 0.0001 for an additional 50 epochs unless early-terminated. Our learning decay of was subtractive 0.0005. Figure 4 shows more generality curves.

Dataset	Training	Testing	Validation	Classes	Training Batch Size
MNIST [17]	50,000	10,000	10,000	10	500
MNIST-random-background [15]	40,000	12,000	10,000	10	500
MNIST-rotated-background [15]	40,000	12,000	10,000	10	500
NIST Special Dataset-19 [9]	271,220	271,220	271,220	62	191
Google Street View House Numbers [19]	63,042	63,042	63,042	10	399
Char 74k English [3]	9,300	3,355	305	62	305
Char 74k Kannada [3]	5,694	1,314	1,752	100	438
MNIST $[4, 5, 8]$	14,000	2,500	2,500	3	500
MNIST $[0, 1, 2, 3, 6, 7, 9] - p$ per-class	$7p$	7,000	7,000	7	500
CIFAR 10 [13]	40,000	10,000	10,000	10	500
Caltech 101 [6]	5,080	3,048	1,016	102	254
Colonoscopy[3]	2,700	900	100	2	100

Table 1. Datasets used and their properties.

4. Results and observations

4.1. Character Datasets

Figure 4 shows the generalities of MNIST-rotated-bg and Kannada prejudiced by all other the character datasets. For reference each plot also shows the generalization performance of a randomly initialized base convolutional network. The following are some observations of interest:

While no dataset is qualitatively the most general, it is quite clear that *MNIST dataset is the most specific*. Rather, MNIST dataset is one that is generalized by all datasets very highly at all layers. Surprisingly, MNIST dataset actually gives better accuracy when prejudiced with other datasets, rather than when initialized with random, if all layers were allowed to learn. This is a strong indicator that *all datasets contain all atomic structures of MNIST*.

NIST, Char74-English and Char74-Kannada follow similar generalization trends with almost all the datasets. With no degrees of freedom they all generalize rather poorly, but their generalities shoot up once one or many layers of the base networks are unfrozen. This indicates two properties: Firstly, these three datasets have similar manifolds. Secondly this also indicates that the last layers of the base datasets are extracting some particular quality of atomic structures that are present in the these datasets alone. Similarly, SVHN does not generalize in the first layer to most datasets, it generalizes much better in the latter layers. This is particularly noticeable in MNIST and Kannada. This further exemplifies the results.

While initially one would have assumed that Kannada would be a general dataset, we observed the contrary. SVHN, Char74-English and Nist generalizes better to Kannada than even Kannada itself does. *English characters seem to be a more general set than Kananda.* While counter-intuitive, this result is immediately obvious when one pays close attention to the filers that are learnt and the dataset itself. Kannada is dominated by predominantly curved edges only, whereas even MNIST has a multitude of unique atomic structures.

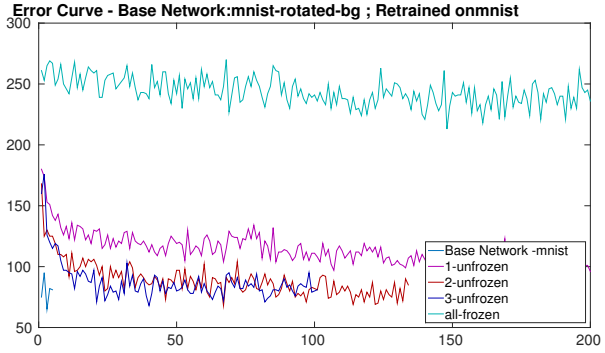

Figure 5. Validation errors vs Epoch number for base-MNIST-rotated-bg retrained on MNIST

Figure 5 demonstrates some interesting phenomenon that we discovered often. *The gain in performance achieved, constantly decreases with increase in degrees of freedom.* Through the epochs, unfreezing only the classifier layer, quickly converges. But while unfreezing, all layers converge at about the same number of epochs. We also observe, that MNIST retrained over MNIST-rotated-background, with *the last degree of freedom does not learn antything at all*. The error rate is within the statistical margin of error. This is a testament to the generality of MNIST-rotated-background among the MNIST datasets. One might expect this because MNIST-rotated-background contains smooth background images (similar to natural image set) and MNIST characters that are rotated. These conditions provide for a good generality.

For the intra-class experiment described in subsection 3.2 above, table 2 shows the accuracies. From the table one can observe that even with one-sample per class,

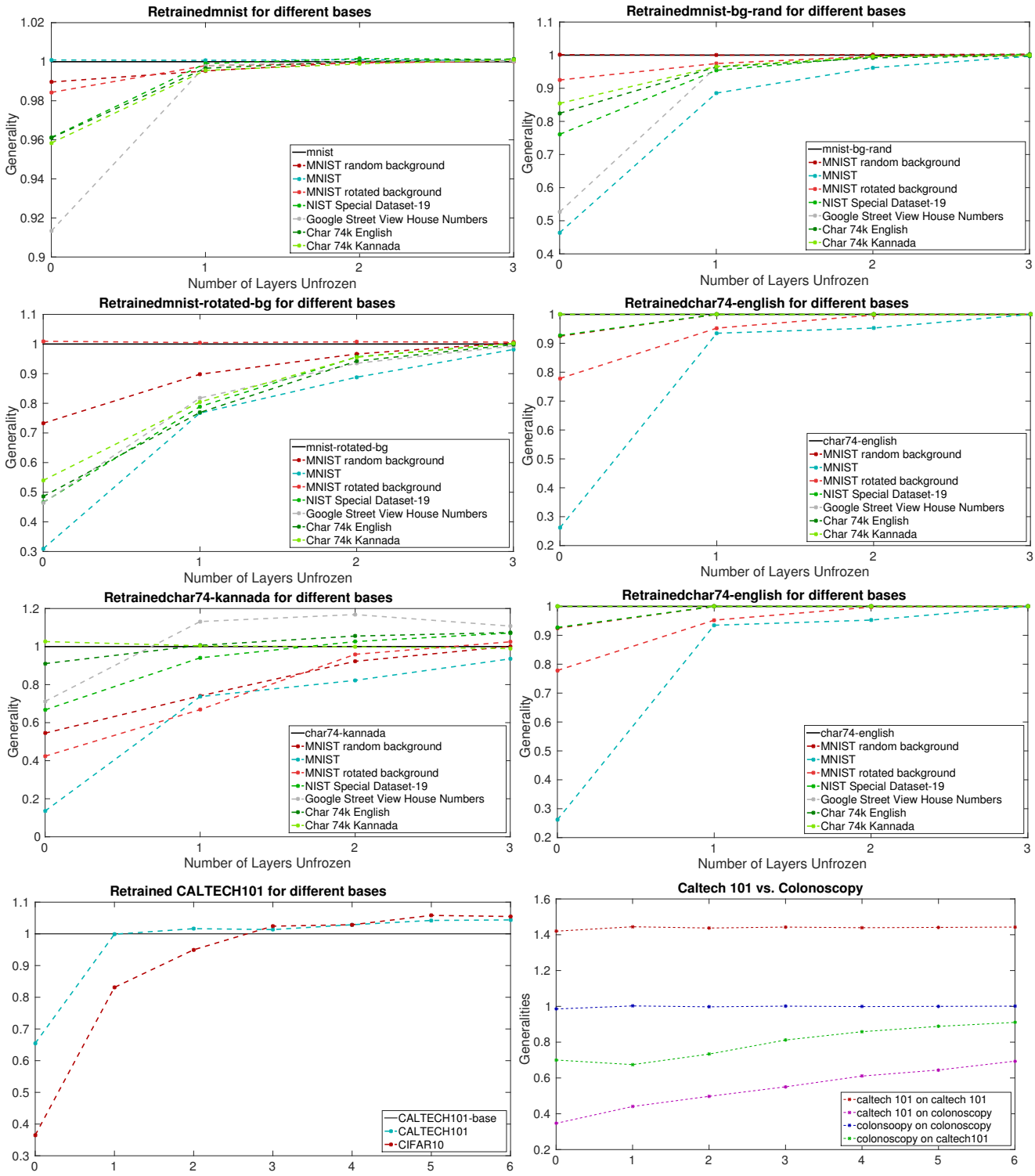

Figure 4. Generalities of datasets not shown in the actual paper. The dark line represents the accuracy of $n(D|r)$. Please zoom on a computer monitor for closer inspection.

a 7-way classifier could achieve 22% more accuracy than a randomly initialized network. It is note worthy that the last row of table 2 still has 100 times less data than the full dataset and it already achieves close to state-of-the-art ac-

p	base	$k=0$	$k=1$	$k=2$	$k=3$
1	Random	-	-	-	55.61
	MNIST[458]	73.07	73.91	76.37	77.52
3	Random	-	-	-	73.34
	MNIST[458]	83.61	87.2	85.7	87.6
5	Random	-	-	-	83.32
	MNIST[458]	90.98	92.98	92.6	92.07
10	Random	-	-	-	81.31
	MNIST[458]	91.55	93.71	93.82	95.08
20	Random	-	-	-	87.77
	MNIST[458]	95.52	95.52	97.07	96.78
30	Random	-	-	-	88.62
	MNIST[458]	96.5	97.34	97.35	97.45
50	Random	-	-	-	90.78
	MNIST[458]	96.38	97.40	97.71	97.38

Table 2. Sub-sample experiment and its generalization accuracies for different layers of freezing. The re-train network was MNIST$[0, 1, 2, 3, 6, 7, 9]$. For obvious reasons random initializations are trained only with all layers unfrozen, hence the missing values.

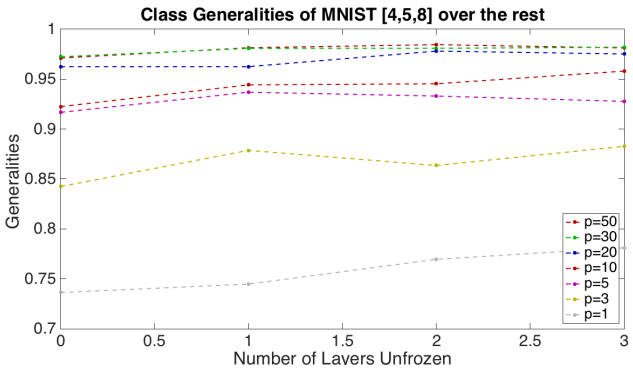

Figure 6. Sub-class generalities for MNIST $[4, 5, 8]$

curacy even when no layer is allowed to change. This is a remarkably strong indicator that the classes $[4, 5, 8]$ generalizes the entire dataset.

Figure 6 mimics the same. We also observed that once initialized with a general enough subset of classes from within the same dataset, the generalities didn't vary among the layers like it did when we initialized with data from outside the mother dataset. We also observed that the more the data we used, more stable the generalities remained. Point of take away from this experiment is that if the classes are general enough, one may now initialize the network with only those classes and then learn the rest of the dataset even with very small number of samples.

4.2. CIFAR 10 vs. Caltech 101

From figure 4 we observe that Caltech 101 doesn't generalize to Cifar 10, which is surprising because Caltech 101 has a lot more classes. One would expect it to be more general. Its quite the opposite because Caltech 101 although has a lot of classes, the variability of each class is not as much as the variability in the Cifar 10 dataset. But it is altogether a serendipitous result that *Cifar 10 is more general than Caltech 101 on the lower layers*. However after three layers of obstination, we find that when the generalities crosses 1, the effect nullifies and reverses slightly. Even though the low-level features are more general in Cifar 10, Caltech 101 generalizes more on higher layers.

4.3. Caltech 101 vs. Colonoscopy

The colonoscopy dataset's labels identify if a image is deemed to be of a quality that is good enough so as to make a diagnosis on the pathology of that particular image. Figure 7 show the filters learnt by Caltech 101 base network and Colonoscopy base network for the exact same architecture from random initialization. Two things are immediately apparent from the learnt filters that while Caltech 101 learns more structured and organized shape features, Colonoscopy dataset learns at first sight what appears to be unstructured blob detectors and detectors for dark colors. These features still produce state-of-the-art accuracy on the dataset. On observation of the activations produced after the first layer, and from observations of images and their labels, one can immediately recognize that what the network is learning is indeed changes in brightness patterns.

Most often the video quality in colonoscopy is affected because of saturation when too much light is thrown at a scene. The quality is also affected due to light reflection from bodily fluids that is also noticeable in the activations. As also can be noticed that most of the filter colors are yellowish or blueish. On an colonoscopy video most often the video is also labelled poor quality when these colors are present, as these colors are often present mostly because of scattering and reflections. Having made these observations one would arrive at the obvious conclusion that neither dataset generalizes the other. This was indeed the result observed from figure 4. Although, Caltech 101 seem to generalize a bit better for even though it predominantly learns shapes, it learns some color features also.

4.4. Summary of results

From all these results and observations, we could summarize that one should prefer to initialize with a general dataset that might have a lot of variability or rather generality in data, when attempting to train with very few number of samples. Whenever possible one must initialize the network trained by a general dataset as this always boosts generalization performance. When there are biased datasets

Figure 7. From left to right, separated by a line are filters learnt by a base Caltech101 base colonoscopy, sample images from the colonoscopy dataset and their first activation for a filter that detects smooth areas of brightness.

with large number of samples in some classes and fewer in others, one should train the most general classes first. Once the network is well-prejudiced one should start introducing the classes with fewer number of and less general samples, provided the general class is general enough.

5. Conclusions

In this paper, we used the performance of CNNs on a dataset when initialized with the filters from other datasets as a tool to measure generality. We proposed a generality metric using these generalization performances. We used the proposed metric to compare popular character recognition datasets and found some interesting patterns and generality assumptions that add to the knowledge-base of these datasets. In particular, we noticed that MNIST data is one of the most specific dataset. We also found that Char74k Kannada is less general than English datasets. We also calculated generality on class-level within a dataset and conclude that a few well-chosen classes used as pre-training could build a network that is well-initialized that even with 100 times less samples, we could learn the other classes. We also provided some practical guidelines for a CNN engineer to adopt. After performing similar experiments on popular imaging datasets and medical datasets, we made similar serendipitous observations.

References

[1] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] Y. N. Dauphin, H. de Vries, J. Chung, and Y. Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390*, 2015. 5

[3] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009. 3, 5, 6

[4] V. Escorcia, J. C. Niebles, and B. Ghanem. On the relationship between visual attributes and convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1256–1264, 2015. 1

[5] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2

[6] L. Fei-Fei, R. Fergus, and 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(1):59–70, 2007. 3, 5, 6

[7] K. Fukushima and N. Wake. Handwritten alphanumeric character recognition by the neocognitron. *Neural Networks, IEEE Transactions on*, 2(3):355–365, 1991. 1

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014. 2

[9] P. J. Grother. NIST Special Database 19 Handprinted Forms and Characters Database. 1995. 6

[10] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 4, 5

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014. 2

[13] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009. 2, 3, 5, 6

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[15] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures

on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007. 3, 5, 6

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 1

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 3, 5, 6

[18] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010. 5

[19] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5. Granada, Spain, 2011. 3, 5, 6

[20] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015. 1

[21] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. 5

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, April 2015. 2

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

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 5

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 1

[26] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014. 2, 4