Low-Shot-Learning of Diseases in Chest X-Rays via Hallucination

Matan Harel Uri Avron Jonathan Somer [*]

September 5, 2018

Abstract

One of the promises of the recent advancements in Artificial Intelligence is the ability to facilitate high precision computer aided diagnosis (CAD) systems and make such high precision diagnosis affordable and highly available. Current methods utilize large labelled datasets in order to achieve high accuracy, but such labelled data is difficult to obtain.

Our objective is to answer whether we can use an existing large set of X-Ray images of healthy patients as well as a large of set of images of patients with some diseases, in order to improve the learning accuracy of new diseases for which we have only a few example images to learn from. This setting is known as Low-Shot-Learning.

We have implemented and extended a method for performing low-shot-learning proposed in [1] and show some novel methods for evaluating this low-shot-learning setting.

1. Introduction

1.1. Low-Shot-Learning

The setting for low-shot-learning is composed of two phases. The first is a representation learning phase: the learner tunes its feature representation on an available set of base classes that have many training instances. In the low-shot learning phase, the learner is exposed to a set of novel classes with only a few examples per class and must learn a classifier over the joint label space of base and novel classes. We evaluate the new classifier's accuracy over both the base and novel classes in order to see that higher accuracy was achieved on the novel classes but also that accuracy was not impaired for the base classes.

Low-Shot-Learning is a great challenge particularly in the medical setting. Patients have varying anatomies, there are different methods of performing each examination, variation might be induced by different equipment and so on. Thus, very large labelled datasets are needed in order to capture this great variation. Obtaining high quality labelled datasets as such is very difficult. In addition, even if labelled data is obtained, the physician's diagnosis can be incorrect and in many cases is not validated or such validation does not get logged. The researchers who gathered the chest X-ray dataset [2] employed a NLP text mining solution to procure labels from physicians written reports, a process which adds further noise to the labelling. Thus, high quality labelled data is difficult to obtain and the ability to learn from little data is highly valuable.

1.2. Our Approach - High Level Glance

Given only few samples of a novel class, we use the abundant data for the base classes to train a generator network which can generate many new 'hallucinated' examples for the novel class. We then use those generated examples, as well as those that were given to us, in order to train another neural network to perform the classification task.

The method for generating new training examples is based on the insight that variation within one category might be transferable to another category. For instance, a certain variation in anatomy may impact the chest images similarly, regardless of the disease.

2. Our Approach

2.1. Datasets used

We tested our results on 3 different datasets. The first 2 are the well researched MNIST and CIFAR10 datasets. The third is a new chest X-ray image dataset. In May 2017, the "ChestX-ray8" dataset, which contains over $100K$ 1024×1024 resolution images, was presented by a team of researchers from the NIH [2]. They presented the methods used to generate the data along with a benchmark for the task of classifying diseases using a deep convolutional neural network (DCNN) they have trained. A succinct sum-

[*] Tel Aviv University, Tel Aviv, Israel. Correspondence to: Uri Avron <uriavron@gmail.com>, Jontahan Somer <somer@mail.tau.ac.il>, Matan Harel <matan.harel.mh@gmail.com>

mary of this dataset is provided in appendix 5.1.

2.2. Phase 1 - Feature Representation Learning

The method we used is composed of 4 phases we describe now. For the X-ray dataset we used a ResNet50 DCNN pre-trained on the very large and diverse ImageNet dataset, without the last dense layer, in order to generate the features for the images. This method was used in [2] in order to train a classifier on the data and implementing the same method enabled us to first recreate their results and continue from there onto low shot learning.

For the MNIST and CIFAR10 datasets we trained two different CNNs which achieved ~99% and ~90% accuracy on the datasets respectively. In the low-shot-learning setting we do not have access to data from the novel class during representation learning. Thus, we treated each class in turn as the novel class , and trained a classifier on the remaining classes. We then used this classifier, with the last layer removed, as a feature extractor in order to generate features for the novel class as well.

Note that there is a significant difference between the two methods. In our method, representation learning is performed ad hoc for the specific setting, intuitively - “we learn to represent digits by learning to recognize digits”. Whereas, in the first setting a generic network is used to generate features for images from a very specific domain. As further research we propose to train a DCNN from scratch on the X-Ray dataset and compare the two methods. We presume that features learnt from data that is close to the domain at hand will be better than those obtained by generic models.

2.3. Phase 2 - Learning to Generate New Examples

We now train a generator G for hallucinating images for novel classes. We train G to “solve analogies”: G will receive as input the concatenated feature vectors $\langle\phi(b_1), \phi(b_2), \phi(x)\rangle$ where b_1, b_2 are two samples from the same base class and x is a novel image. For this input, G will output a vector who solves the analogy $b_1 : b_2 \Rightarrow x : ?$ Thus applying to x the $b_1 \rightarrow b_2$ transformation. Note that the $b_1 \rightarrow b_2$ transformation stays within class B and the generator should perform on x a transformation that does not result in an element of a different class than x (see part 3.3 for further evaluation of the generator’s performance).

G will be a 3 layer MLP. The training data for G is generated by creating completed analogies - quadruplets of feature vectors from the base classes. We start by clustering each of the base classes into k clusters (we tested different k ranging from 5 to 50). Then for each two classes A and B, for each pair of centroids, c_1^A, c_2^A from class A we find the pair c_1^B, c_2^B such that the cosine distance between $c_1^A - c_2^A$ and $c_1^B - c_2^B$ is minimized. Concatenating these 4 centroids results in one element in the dataset. For each quadruplet $\langle c_1^A, c_2^A, c_1^B, c_2^B\rangle$, we feed the triplet: $\langle c_1^A, c_2^A, c_1^B\rangle$ into G. We want $G(\langle c_1^A, c_2^A, c_1^B\rangle)$ to be as close as possible to c_2^B and also remain within the class B. In order to do so we minimize the loss function:

$$\lambda MSE(G(\langle c_1^A, c_2^A, c_1^B\rangle), c_2^B) \\ +(1-\lambda)L_{cls_{BASE}}(G(\langle c_1^A, c_2^A, c_1^B\rangle), B)$$

Where we have MSE the mean square error between the generator’s output and the true target and $L_{cls_{BASE}}$ the log-loss of the classifier w.r.t the true class of c_2^B.

2.4. Phase 3 - Generating New Examples for a Novel Class

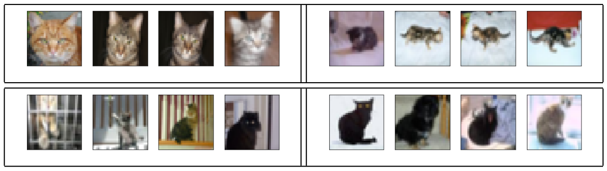
Assume we have received n examples of some novel category. Generating a new example is done by sampling one of these examples - $\phi(x)$, choosing a base class A , and from it two centroids - c_1^A, c_2^A. We then apply the generator to this triplet; $G(\langle c_1^A, c_2^A, \phi(x)\rangle)$ is the generated example.

In [1], the choice of category and centroids was performed uniformly at random. Recall that our goal is to compensate for the lack of intra-class variation contained within the samples of the novel class. Thus, we would like to generate this variation utilizing information about other classes. We propose a novel way of performing this selection which increases the likelihood that the variation within the chosen base class A will be transferable to the novel class. Recall that during the feature representation phase we have trained a classifier over the base classes. We suggest that by applying this classifier on the samples of the novel class, we will gain some insight into which classes might possibly carry greater transferability to the novel class. For instance, applying the CIFAR classifier to predict the class of a “cat” example results in “dog” in many cases. It is very likely that the variation in the dog class is much more relevant to the cat class than for instance the variation in the airplane class, as dogs and cats should appear in similar conditions whereas airplanes and cats should not.

2.5. Phase 4 - Training a Classifier on Generated Data

The main consideration here is that the low-shot setting creates a significant class imbalance. In order to fix this we follow [1] and sample the training data uniformly over classes and uniformly within classes.

Figure 1. Clusters in Feature Space Capture Intra-class Variation



3. Experiments

3.1. Evaluating Clustering over Feature Space

One of the assumptions behind this model is that clustering over feature space results in capturing the variation in the data. We used the K-Means algorithm in order to cluster the data. Recall that K-Means produces centroids that are not necessarily in the original set of points. So in order to test if these centroids capture a variation in the data we first found for each centroid the 4 features closest to it. We then plotted the 4 images whose feature representations are these 4 examples. Figures 1 and 6 validate the assumption that the clustering grasps the intra class variation. Note that the total number of clusters was 30 and we randomly chose 4 to display.

3.2. Connection Between Number of Clusters Used in Training and Test Accuracy Achieved

Our generator is trained to perform transformations which move "one centroid to another". There is an underlying assumption here that we must have data from different clusters in order to capture the variation and perform well on the test data. We wanted to test this assumption before testing the use of generated data. We first plot for each dataset the accuracy achieved by the number of unique samples used for training. Figure 5 shows that the growth is logarithmic in the number of samples. We now fix the sample size and test how the number of clusters from which the training data is taken affects the accuracy. We chose to use 256 samples as we have seen that the growth in accuracy diminishes from that point forward. Figure 2 shows that taking data from more centroids does indeed increase the model's capacity to generalize on the test set. All results are averaged over 5 runs.

3.3. Evaluating the Performance of the Generator

3.3.1. Task 1: Transformations Generate new Instances of the Same Class

In essence, the generator must perform well on two distinct tasks. The first: given a sample from a novel class the generator must generate samples which are from the same class. This is non-trivial since it has never trained on data from this class. There is an underlying assumption that training the generator to perform transformations which leave examples from the base classes in the same class will result in the generator performing similarly on the novel class. As the generated examples do not correspond with real images, we devised the following test to check if they are from the correct class: we first train a classifier on all examples of the base and novel classes, this classifier serves as our "oracle". We expect that this classifier will recognize the generated images from samples of the novel class as instances of the novel class, with high probability.

Figure 2. Accuracy Achieved by Number of Clusters Data is Sampled From - CIFAR (Fixed Number of Unique Samples)

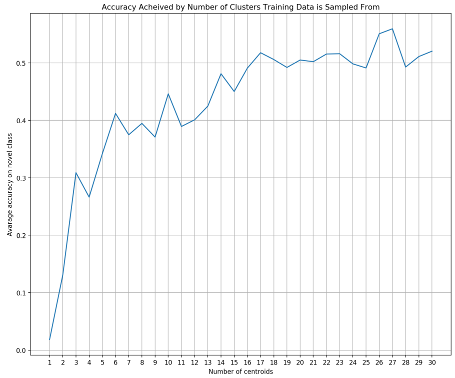
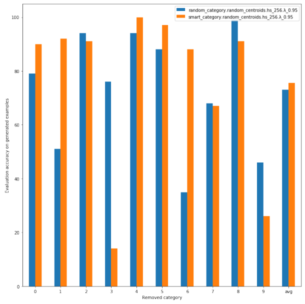


Figure 3. Accuracy Acheived on Generated Data + Comparison of Methods - MNIST



At this point we also provide a comparison between the standard method of generating data - choosing a category at random, and our "smart selection" of a category. We show that this new method provides some improvement over the standard method of generation. Figure 3 plots this comparison, our method in orange. The X axis corresponds with the class removed while training the generator - this is the novel class. Note the rightmost column which depicts the average accuracy over all categories. We also provide the parameters we used: The number of neurons used for the generators hidden layers: 256, and the parameter $\lambda = 0.95$ used in the loss function. These were discovered via cross validation. We fix these parameters from here on.

3.3.2. Task 2: Samples Generated have Significant Variation

Notice that this first task could be achieved trivially by returning the sample received as it is. If the “oracle” classifier recognizes this sample to be from its correct class, this trivial method will achieve 100% accuracy in the previous test we provided. This will not provide for any additional capacity for generalization.
Thus, we must also evaluate the generators ability to induce variation in the generated samples. We do this by first clustering all of the novel class’s data (not only the samples given to us). We now perform the following test, using the MNIST dataset: from each of the clusters we draw a single sample, we then generate 256 new samples from it using our generator. We then count the number of clusters these generated examples came from. As a sanity check we also re-evaluated the accuracy achieved by the "oracle" classifier and ensured that it was sufficiently high as before.

The average number of unique clusters the data came from was 5.84 for the standard method of random category selection and 5.62 for our “smart” category selection. We conclude that the generator does indeed supply variance in the generated data.

3.3.3. Comparing the Generator’s Performance on the Different Datasets

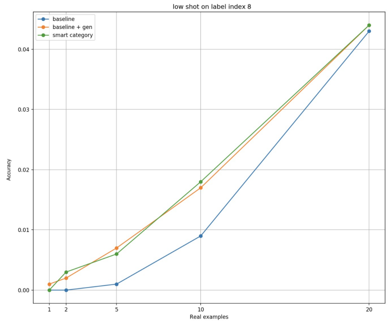
At this point, after testing different sets of hyper-parameters, we could not arrive at significant accuracy on generated samples from the X-ray dataset. Furthermore, we have noticed that the generator trained on the X-ray dataset generates the same point no matter what input it receives. We attribute this to the generator’s lack of ability to discover any underlying patterns in this noisy data, at which point it “settled” for some mid-point which minimizes expected loss. Note that our architecture was able to achieve high accuracies with both the MNIST dataset and the CIFAR10 dataset.

We attribute this difference to the noisy labels of the X-ray dataset and the size and complexity of its images. Testing this, we attempted to train a DCNN on this data but could not achieve an accuracy higher than 25%. We conclude that achieving a sufficient accuracy on the base categories is a pre-requisite to performing low-shot learning on a dataset as training the generator using the loss from a weak classifier simply adds noise to the training.

3.4. Training A Classifier On Generated Data

We first perform the following experiment from [1]: for each n in $\{1, 2, 5, 10, 20\}$ we generate $20 - n$ new samples and plot the accuracy achieved. We were able to recreate the results achieved in [1] - improvement for small n (in [1] only for $n = 1, 2$). Figure 4 shows the results from [1].

Figure 4. Accuracy Over number of Original Samples Given



Note that in the paper they have used top-5 accuracy so the exact accuracy we achieved is in a different order of magnitude. Figure 7 shows the results of this experiment for a single class from CIFAR10.

We decided to implement yet another test. What we believe to be the most significant for a production setting is achieving the maximum accuracy possible for a given constant number of examples given. We plotted this exactly, using a constant number of 5 examples. Figure 8 in the appendix shows the results. We see an initial increase using both methods, followed by a decrease. We believe that the generated data is noisy and from some amount of generated samples, the proportion of true samples to generated samples becomes such that accuracy is impaired.

4. Conclusion

We started this project with the goal of achieving high precision low-shot learning on a large-scale, high resolution medical dataset. As we progressed we have found that our focus moved to coming up with novel ways to evaluate and visualize the method proposed for low-shot-learning. Given the results we have achieved we conclude that many of the underlying assumptions for this model are indeed correct. We suggest considering alternative architectures for the generator as it fails to generate examples that can be used to achieve high accuracies.

One of the most interesting insights we have arrived at is that there exists a significant gap between data which is recognizable as part of some class and data that can be used to learn this class from. We have seen that the generated examples can be recognized as elements of a class with great consistency, but a new classifier, trained on these instances does not perform well.

References

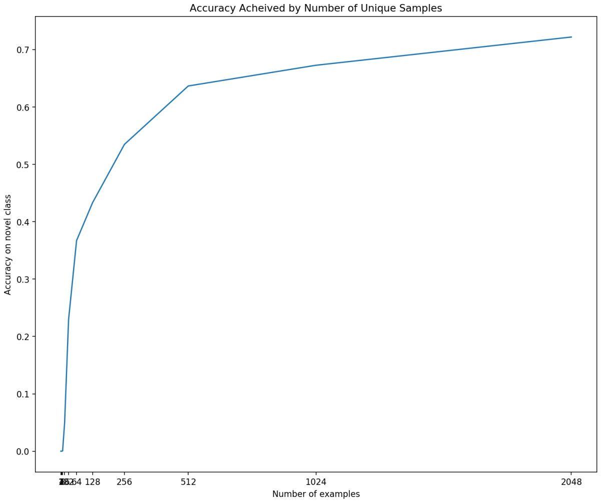
[1] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. *arXiv*, 2016.

[2] Xiaosong Wang. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks. *National Institutes of Health*, 2017.

5. Appendix

5.1. Accuracy as Function of the Number of Samples used For Training

Figure 5. Accuracy Achieved by Number of Unique Samples Trained On - CIFAR



5.2. Chest X-Ray8 Dataset Summary

- 108,948 images of 32,717 patients.
- 8 disease labels text-mined from radiological reports.
- Each image is labeled with ’Normal’ or labelled with more of the 8 disease labels, or
- Labeling: classes are very imbalanced. For example: 84K images were tagged ’Normal’ and around 1K were tagged with ’Cardiomegaly ’.
- Image sizes are 1024 × 1024 pixels. These are relatively large images, recall CIFAR images are 32×32. The entire dataset takes around 40GB of space.

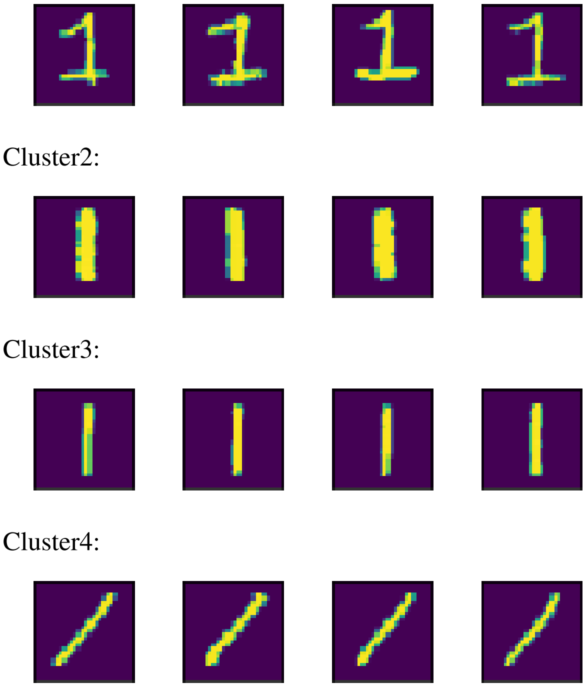
5.3. Resources

We utilized 3 google cloud instances, equipped with Nvidia Tesla V100 GPUs. This setup enabled us to parallelize cross validation and brought down computation time from days to hours for some tasks.

5.4. Clustering Over Feature Space - Results for MNIST

Figure 6. Clustering over Feature Space Captures Intra-class Variation - Digits

Cluster1:



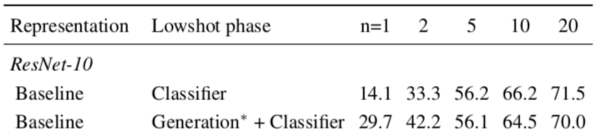
Cluster2:

Cluster3:

Cluster4:

5.5. Results From the Original Paper

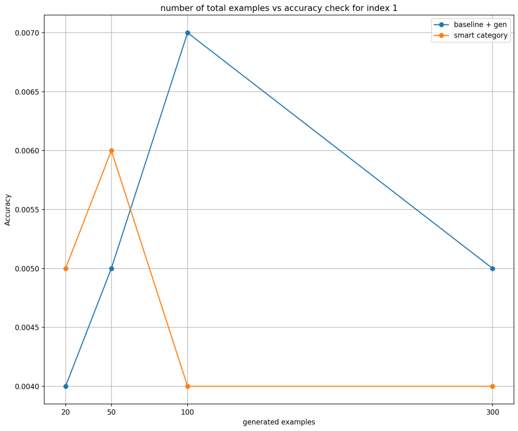
Figure 7. Accuracy for Novel Class by Number of Samples Given



Representation	Lowshot phase	n=1	2	5	10	20
ResNet-10						
Baseline	Classifier	14.1	33.3	56.2	66.2	71.5
Baseline	Generation* + Classifier	29.7	42.2	56.1	64.5	70.0

5.6. Accuracy by Number of Examples Generated

Figure 8. Accuracy for Novel Class by Number of Samples Generated



5.7. Code For the Project

https://github.com/JonathanSomer/AdvancedMachineLearningCourse/tree/master/project/code`https://github.com/JonathanSomer/AdvancedMachineLearningCourse/tree/master/project/code`