A Semi-Supervised Approach for Image Labeling with Affinity Coding

Ryan Cooper[†]
Georgia Institute of Technology
Atlanta, Georgia
ryan.cooper@gatech.edu

Dharmashloka Debashis[†]
Georgia Institute of Technology
Atlanta, Georgia
shlok@gatech.edu

ABSTRACT

Modern machine learning techniques, especially in the domain of images owe much of their success to large quantities of labelled training data. However, they are not easy to obtain and hence not scalable. There is a need to automatically label datasets in an unsupervised manner by making use of only a small labelled set. GOGGLES[1] does that by proposing a novel way to obtain probabilistic labels for unlabelled image datasets automatically. They make use of pretrained networks to extract features at multiple scales and use similarities between images to cluster them. Then they assign labels using a small labelled set. We challenge various aspects of GOGGLES by replacing its core mechanisms with our approaches. In particular, we replace their hierarchical clustering method with a GAN Mixture Model. We also involve the development set to obtain the features for clustering by training an embedding layer. This led to a boost in accuracy by upto 18.6% from GOGGLES for the CUB dataset. We also added a mechanism to improve the accuracy with increase in the size of the labeled set, which was missing in GOGGLES. Our code can be found at https://github.gatech.edu/ddebashis3/dmm-project-fall2020

1 INTRODUCTION

One of the major limitations in modern machine learning and deep learning tasks as of late has been the lack of labelled training examples. It is often the case in specific-domain and organizational settings, there is ample amount of unlabelled data, and it is often infeasible for all of this data to be labelled for a given task creating unique challenges for leveraging machine learning in these settings. Techniques like few-shot learning, transfer learning, and semi-supervised techniques have been shown to be effective in labelling large corpuses of data with only a few examples.

Of the approaches that currently exist in the data management community, one of the more promising branches is the Data Programming paradigm, which seeks to reduce the human effort in labeling [2] [3]. In Data Programming, domain experts write labeling functions, which describe the classes with annotated information, such as text or metadata; but does not extend as easily to domains where semantics are not explicit since expressing domain knowledge directly using these domains' raw features is difficult.

1.1 Limitations of Current Approaches

One of the newest techniques in the image domain in Data Programming is GOGGLES [1] (the detailed architecture is discussed in section 2.4), a solution which uses affinity functions with a generative model for class labeling over images, which exploits weak supervision as part of the labeling scheme. They pose their framework as a cluster-then-label approach for few-shot learning, where the general intuition is that a pair of images belonging to the same class are more similar (according to some similarity function) than the images belonging to different classes.

The authors pose the design of affinity functions to be domain-agnostic, using pre-trained convolutional models and extracting some latent representation from middle layers. This allows them to design a class of affinity functions that is able to capture image semantics without the need for a labelled dataset. The quality of the affinity functions capturing the image semantics is clearly extremely important to the labelling quality. This raises the concern that the quality of the labelling is dependent on the quality of the affinity functions to capture semantics, raising the question of the quality of the affinity function design. They address this by using affinity functions with weak classifiers, but it may be possible to capture more complex semantics in other manners.

Though the technique proposed in GOGGLES has been shown to be extremely efficient in labelling with very few samples and the results showed that the proposed method saturates to some maximum accuracy regardless of the size of the development set and the number of affinity functions used. While this is good for proving the power of this technique, it would be beneficial if the accuracy of the technique increases given more data as is typical in other data labelling methods.

The use of affinity functions to capture image semantics in the data programming paradigm is a relatively new idea, and exploring the design of these functions could be useful in understanding how this methodology could be sensitive to varying different components.

1.2 Proposed Work

From the limitations we see in GOGGLES and other works in unsupervised data labelling, we arrive at the following research questions:

(1) The quality of the labelling is dependent on the quality of the affinity functions to capture semantics. Maybe there is a more robust way to design these functions to capture latent semantics?
(2) Would including the development set as part of the affinity function generation improve the quality of the affinity functions, or would it lead to overfitting? Can the addition of more training data lead to higher labelling accuracy rather than just converging?

[†] These two authors contributed equally.

A useful side effect of many machine learning models is that increasing the amount of training data consequently increases the training accuracy of the model. This is not the case in GOGGLES. This could potentially lead to overfitting if done incorrectly as the training corpus is small.

(3) Are there models that could outperform the hierarchical generative model in high dimensional clustering?
Aside from the design of the affinity functions themselves, one of the critical points of focus in the GOGGLES paper was the use of GMMs in the base model, and a Multivariate Bernoulli distribution is fitted using EM in the ensemble step (hierarchical generative model). Given the nature of high dimensionality of the space, it has been shown the GAN mixture models (GANMM) and other types of GAN based solutions are able to outperform GMMs on clustering of high dimensional and complex data (more on this in Section 2.3. This leads to the natural conclusion that a more complex model may be more suitable to this application.

2 RELATED WORK

2.1 Few-shot Learning

The area of few-shot learning claims that it is possible to train a classifier using a very limited amount of data. The major approaches can be categorized into meta-learning and self-supervised learning.

Meta-learning means learning how to learn. The idea here is to prepare a network to generalize well in a new task with limited data by training it first on one or more supervised tasks. J. Snell et al. [4] do this by introducing episodic learning where they mimic the test scenario by training on a different subset task on each iteration of meta-training and making use of average activations of images belonging to the same class as prototypes for vector quantization of the classification space. S. Qiao et al. [5] take this approach further by training a meta-network on top of a pre-trained backbone to generate parameters for the new task from the activations of the meta-network. Probabilistic class-labels are obtained from a softmax layer which is added after taking the inner product of the parameters (which are the activations of the meta-network) and the activations from the backbone.

Self-supervised learning is an approach based on using a large corpus of unsupervised data for training the CNN backbone of a network to enrich the feature representation for downstream tasks. This direction is well-represented by the work of D. Chen et al. [6] where they train the backbone by treating each individual image as one class. They take multiple augmented versions of an image and comparing it with other images by taking features from multiple feature-maps from the backbone to minimize the Noise Contrastive Estimation (NCE) loss.

2.2 Deep Metric Learning

The goal of deep metric learning is to use deep neural networks to learn an embedding space where the distance between a pair of samples is commensurate with their semantic visual similarity. In other words, the aim is to train a similarity metric for images. This field of research is very relevant to us as the affinity functions in GOGGLES we will be working on are just another name for similarity functions.

In the literature, the most common kind of loss function used for training an embedding layer on a supervised classification task is called a 'triplet-loss' [7] and in its more general form, the 'N-pair loss' [8–11] . The triplet loss works by choosing an image from the dataset as an anchor, then choosing one positive image (same class as the anchor) and one negative image (different class from the anchor). The loss is them computed by maximizing the anchor-to-negative distance and minimizing the anchor-to-positive distance. This process is iterated over a set of triplets mined from the dataset. F. Schroff et al. introduced the triplet loss for tackling the face recognition problem with limited data. To take care of computational feasibility of the N-pair loss [11], Y. Movshovitz-Attias et al. [9] designed the 'Proxy-NCA' loss and represented each class with a learnable proxy and used these proxies for positive and negative samples given an image as an anchor while computing the N-pair loss. However, replacing images with their proxies prevents the model to learn direct data-to-data relations as in the case of the vanilla N-pair loss. To solve this problem, S. Kim et al. [8] design the 'Proxy-Anchor' loss function that uses the proxies as anchors and uses images directly for positive and negative samples.

We plan to take inspiration from these few-shot learning and deep metric learning approaches to train an embedding layer on the development set and use the activations from this embedding layer to compute new affinity functions for clustering using GOGGLES. Since these affinity functions use the labeled development set, they will be more informative than the other affinity functions of GOGGLES that are obtained in a completely unsupervised manner. Hence, it will help guide the clustering process to converge to a better optimum which is aligned with the task at hand.

2.3 GANs

Generative Adversarial Networks (GANs) are class of machine learning framework rooted in the image domain that have been previously used in addressing the image construction and they are able to capture complex data distributions from raw feature spaces. GANs train a generative network, aimed to generate data indistinguishable from the training data, and a discriminative network, aimed to separate the training data set from the generated data, simultaneously. This adversarial process converges such that the generated data has a distribution close to the training data. Although GANs were not a mechanism originally designed to cluster, two prominent models architectures have recently been published; ClusterGAN [12] and GANMM [13].

Research by Zhou and Yu [13] has shown that GANs are competitive mechanisms compared to popular GMMs in high dimensional clustering. Since GANs make no assumptions about the underlying distribution in the mixture model, they are able to model complex data distributions even from raw features. This is a promising finding as they show that through ϵ-expectation-maximization (ϵ-EM), they can learn the complex data distribution on a cluster-by-cluster basis through an EM methodology, refer to Figure 4 for this architecture. The exact details of this implementation are not included here as the main focus of this technology is the theoretical applications in high dimensional clustering. ClusterGAN solves the same problem but in this approach, the latent space between the networks is

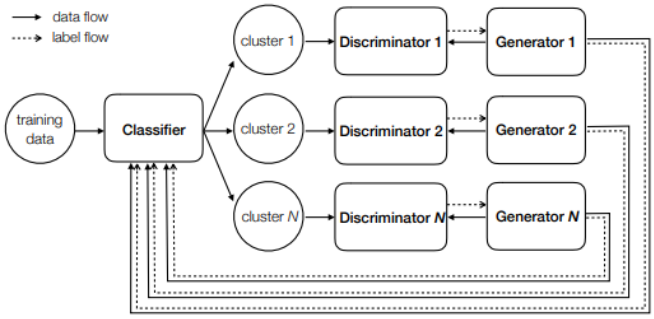

Figure 1: GANMM cluster architecture [13]

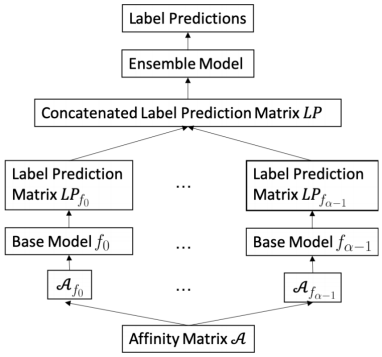

Figure 2: GOGGLES architecture From [1]

sampled using a mixture of discrete and continuous latent variables in order to create a non-smooth geometry in the latent space.

2.4 GOGGLES

GOGGLES takes a cluster-then-label approach to the data programming paradigm by using labeling functions that are designed to capture the inherent semantic features in an image. GOGGLES can be thought of as a workflow, where given some new dataset $\mathcal{X}$. These steps can be visualized by Figure 2.

(1) For every max pooling layer in VGG-16 extract the prototypes (semantic concepts) and compute the pairwise similarity for all samples in the unlabelled corpus. This results in a matrix of the size $\mathcal{A} \in \mathbb{R}^{(n+m)\times\alpha(n+m)}$.
(2) Given $\mathcal{A}_f \in \mathbb{R}^{N\times N}$ generated by some affinity function, then the class probability can be learned through GMM.
(3) Concatenate the output from the GMM base models into a single Label Prediction (LP) matrix, then discretize the concatenated matrix and fit a Multivariate Bernoulli distribution using EM to get final label predictions.

These steps summarize the general workflow of GOGGLES, with the exact implementation details found in [1].

3 FORMAL PROBLEM DEFINITION

We have an unlabeled set of images $S = \{s_1, s_2, \ldots, s_N\}$, and a development set $D = \{(x_1, y_1), (x_2, y_2), ..., (x_M, y_M)\}$, where $\{x_1, x_2, .., x_M\} \subset S$ and $y_i \in \{0, 1, \ldots, K\}$. The development set is very small as compared to the set S, i.e., $M \ll N$. The goal is to learn parameters θ of a classifier C_θ using the unlabeled set S and the small labeled set D which can assign probabilistic labels $\tilde{y_i} = \{\tilde{y_i}^0, \tilde{y_i}^1, ..., \tilde{y_i}^K\}$ to each of the elements $s_i \in S$, where $\tilde{y_i}^k = \Pr(\tilde{y_i} = k), k \in \{0, 1, \ldots, K\}$. In our experiments, $K = 2$. The hard labels $\hat{y_i} \in \{0, 1, \ldots, K\}$ can be obtained by taking the index corresponding to the highest probability, that is, $\hat{y_i} = \underset{k}{\text{argmin}}\{\tilde{y_i}^k\}$.

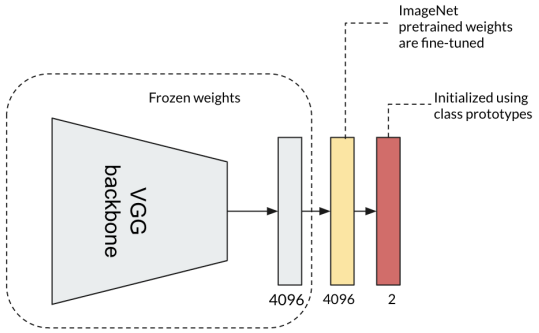

Figure 3: Schematic for the cross-entropy-based training of embedding layer

These probabilistic labels (or the hard labels) can be used to train a downstream classifier model. However, this is out of the scope of this paper. We are only interested in preparing a labeled dataset in a semi-supervised manner.

4 METHODOLOGY

4.1 Datasets

We make use of the following datasets:

(1) **CUB**[14]: A total of 11,788 bird images from 200 species. We use the a similar sampling strategy as GOGGLES [1] for evaluating binary classification results. However, GOGGLES was not clear on the exact class-pairs that they used for binary classification. For the experiments in this paper, we categorized pairs of classes into three types based on the labelling accuracy obtained using GOGGLES. The categories are: CUB Easy ($\geq$ 96% accuracy), CUB Moderate ([88 − 96)% accuracy), and CUB Hard ($<$ 88% accuracy) as listed in Table 1. We chose 10 pairs for each of those categories. We include the exact names of the classes for replicability in future work. This dataset is in-domain, as the images are very similar to the Imagenet dataset which was used to train the VGG-16 feature-extractor used in this work.
(2) **TB-Xray**[15]: It contains 662 images of chest X-ray images which are either 'normal' or 'abnormal' where 'abnormal' shows signs of tuberoculosis. This dataset is out-of-domain as X-ray images are very different from the natural images of Imagenet.

CUB Easy (>=96% accuracy on Goggles)	CUB Moderate (88-96% accuracy on Goggles)	CUB Hard (<88% accuracy on Goggles)
001.Black_footed_Albatross, 007.Parakeet_Auklet	154.Red_eyed_Vireo, 109.American_Redstart	017.Cardinal, 056.Pine_Grosbeak
095.Baltimore_Oriole, 036.Northern_Flicker	023.Brandt_Cormorant, 026.Bronzed_Cowbird	158.Bay_breasted_Warbler, 037.Acadian_Flycatcher
070.Green_Violetear, 030.Fish_Crow	065.Slaty_backed_Gull, 026.Bronzed_Cowbird	011.Rusty_Blackbird, 195.Carolina_Wren
015.Lazuli_Bunting, 070.Green_Violetear	137.Cliff_Swallow, 026.Bronzed_Cowbird	104.American_Pipit, 126.Nelson_Sharp_tailed_Sparrow
102.Western_Wood_Pewee, 002.Laysan_Albatross	156.White_eyed_Vireo, 161.Blue_winged_Warbler	173.Orange_crowned_Warbler, 031.Black_billed_Cuckoo
034.Gray_crowned_Rosy_Finch, 038.Great_Crested_Flycatcher	156.White_eyed_Vireo, 039.Least_Flycatcher	126.Nelson_Sharp_tailed_Sparrow, 037.Acadian_Flycatcher
036.Northern_Flicker, 074.Florida_Jay	029.American_Crow, 037.Acadian_Flycatcher	178.Swainson_Warbler, 163.Cape_May_Warbler
010.Red_winged_Blackbird, 033.Yellow_billed_Cuckoo	165.Chestnut_sided_Warbler, 130.Tree_Sparrow	163.Cape_May_Warbler, 088.Western_Meadowlark
041.Scissor_tailed_Flycatcher, 026.Bronzed_Cowbird	031.Black_billed_Cuckoo, 098.Scott_Oriole	163.Cape_May_Warbler, 165.Chestnut_sided_Warbler
166.Golden_winged_Warbler, 167.Hooded_Warbler	072.Pomarine_Jaeger, 029.American_Crow	180.Wilson_Warbler, 177.Prothonotary_Warbler

Table 1: Table showing the class breakdowns for CUB Easy, Moderate, and Hard

4.2 New Affinity functions

In this section, we discuss the various ways in which we obtain new kinds of affinity functions, which we then use to obtain probabilistic labels using the same hierarchical generative model of GOGGLES.

4.2.1 Using Fully-connected layers. The affinity functions of GOGGLES are obtained using the the 2D feature-map layers from the VGG-16 pretrained network. As we will see in the ablation studies, the top layers are as much informative, and sometimes more informative than the bottom layers. Hence, we extract features from the two fully connected layers (before the classification layer) to construct affinity functions. We also extract features from the pre-classifier fully-connected layer from pre-trained Inception-V3 and Resnet-50 networks to construct affinity functions.

4.2.2 Learning Embedding layer using the Development set. It is desirable that with increase in the size of the labelled training set, the accuracy should increase, which is not true for GOGGLES. With that objective, we make use of the development set to train an embedding layer on top of the VGG-16 pretrained network and extract features from that layer to design affinity functions. Since the development set is involved in generating those affinity functions, it would directly impact the clustering of the unlabelled set. We use two different approaches to train that embedding layer.

In the first approach (Fig. 3), we freeze the pre-trained VGG-16 backbone up to the first fully-connected (FC) layer. The second FC layer is initialized using the VGG-16 pretrained network but not frozen. The classifier layer is initialized by using class prototypes. Class prototypes are obtained by taking the average of the activations from the previous pre-trained layer of all the images in that class in the development set. So, we initialize the parameters using the activations of the previous layer. The intuition behind that is, since we have a very small number of training data, a good initialization is important so as to not overfit on that data. We then finetune the last two FC layers on the development set using cross-entropy loss.

For the second approach, we initialize in a similar way but use a triplet-loss instead of the cross-entropy loss to train the embedding layer.

4.3 Clustering using GANs

We utilize the clustering technique described in [13] with the $\epsilon-$expectation method described for convergence. We adapt a generator and discriminator dynamically based on the size of the affinity matrix. With these modifications, the GANMM code can directly be used to cluster then label. This approach should be implemented in such a way that it is invariant to the input dataset and fully adaptable. We play to test this on the three CUB datasets we have defined and TB-Xray for robustness.

5 EXPERIMENTS

5.1 Experiments on the CUB dataset

Table 2 shows labeling accuracies for with our new affinity functions, with the same clustering method as GOGGLES.

All_AFs are the original 50 affinity functions from GOGGLES. ***first40***, ***last10*** are respectively, the accuracies obtained by using the first 40 affinity functions and last 10 affinity function from the original GOGGLES set. ***AF40, AF49***, are respectively the 41^{st} and 50^{th} affinity function.

vgg_fc1, ***vgg_fc1*** are the respectively features from the first and second fully-connected (FC) layer from VGG-16. ***invc3*** is the pre-classifier FC layer from the pretrained Inception-V3 network. Similarly, ***res50*** is for the pretrained Resnet-50 network. ***vgg-ft*** is obtained using the embedding layer trained on the development set using cross-entropy loss.

Table 3, 4, 5 show the results of using the same 50 affinity functions from GOGGLES but using GAN Mixture Models (GANMM) for clustering.

Table 6 shows the effect of combining the embedding-layer-based affinity function with GANMM clustering.

5.2 Experiments on the TB-Xray dataset

Table 7 summarizes the experiments performed on the TB-Xray dataset. ***AF48*** is the most informative affinity function that gave the highest accuracy among the GOGGLES affinity functions. ***Triplet AF*** is obtained from the embedding layer using the triplet-loss. ***vgg-ft*** is obtained using the embedding layer trained using cross-entropy loss. ***GAN*** uses GOGGLES' 50 affinity functions and clusters using GAN Mixture Models (GANMM). ***vgg-ft + GAN*** uses the ***vgg-ft*** affinity function and clusters using GANMM.

	Easy	Moderate	Hard
All_AFs	98.3256184	91.851604	67.6568888
first40	89.7786878	85.895022	63.6477611
last10	98.4901605	91.240196	71.4566258
AF40	97.7345582	90.478419	77.1709709
AF49	98.154026	91.240896	70.6225804
vgg_fc1+vgg_fc2	97.8967748	89.160364	78.1663717
vgg_fc1	97.4786957	88.407563	78.5930219
vgg_fc2	95.3757664	90.242297	76.8266765
incv3+res50	93.7746777	87.101795	70.1367492
incv3	88.9889515	86.776038	67.9483014
res50	94.1993077	86.886809	70.3371931
vgg-ft	97.0609006	95.908263	79.5637726

Table 2: Table of experimental results varying the Affinity Functions over CUB datasets

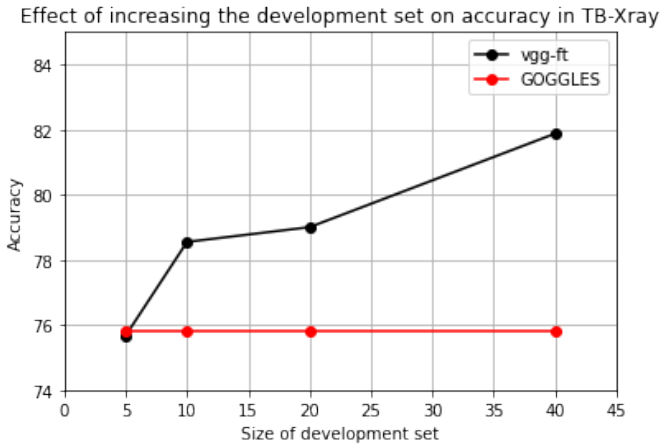

Figure 4: Comparing the effects of increasing the development set size using vgg-ft affinity functions versus GOGGLES affinity functions on TB-Xray

Label Pair	Hierarchical Accuracy	GAN Accuracy	Rel. Improvement
001, 007	100.0000	100.0000	0.0000
095, 036	100.0000	100.0000	0.0000
070, 030	99.1666	100.0000	0.8404
015, 070	99.1525	99.1525	0.0000
102, 002	98.3333	99.1666	0.8474
034, 038	98.3193	98.3193	0.0000
036, 074	97.5000	98.3333	0.8547
010, 033	97.4789	98.3193	0.8622
041, 026	96.6666	98.3333	1.7242
166, 167	93.2773	96.6386	3.6036
AVERAGE	97.9895	98.8263	0.8732

Table 3: Experimental results over CUB Easy using GANMM instead of hierarchical generative model

Label Pair	Hierarchical Accuracy	GANMM Accuracy	Rel. Improvement
154, 109	95.0000	95.0000	0.0000
023, 026	94.1176	96.6386	2.6786
065, 026	95.4545	96.3636	0.9524
137, 026	93.3333	96.6666	3.5714
156, 161	93.3333	93.3333	0.0000
156, 039	91.5966	94.1176	2.7523
029, 037	90.7563	89.0756	-1.8519
165, 130	85.0000	85.0000	0.0000
031, 098	91.6666	95.0000	3.6364
072, 029	88.3333	90.8333	2.8302
AVERAGE	91.8592	93.2029	1.4569

Table 4: Experimental results over CUB Moderate using GANMM instead of hierarchical generative model

Label Pair	Hierarchical Accuracy	GANMM Accuracy	Rel. Improvement
017, 056	90.5983	91.4530	0.9434
158, 037	77.3109	85.7142	10.8695
011, 195	75.0000	80.0000	6.6667
104, 126	72.2689	81.5126	12.7907
173, 031	94.1667	96.6666	2.6548
126, 037	34.7458	55.0847	58.5366
178, 163	68.1034	72.4137	6.3290
163, 088	57.5000	63.3333	10.1449
163, 165	54.1667	57.5000	6.1539
180, 177	46.6667	55.8333	19.6428
AVERAGE	67.0527	73.9511	13.4732

Table 5: Experimental results over CUB Hard using GANMM instead of hierarchical generative model

6 DISCUSSIONS & ABLATION STUDY

From Table 2, we can make a number of conlusions. First, a small number of affinity functions are enough, and sometimes work better than having a large number of them. Second, The last few affinity functions (at the top) are more informative than first several affinity functions. Third, using the development set to train the embedding layer improves the accuracy by a lot in case of hard pairs of classes. This might be because the clustering is better guided when we use the information from the labels of that dataset while clustering.

In Table 3 we observe that GAN and Hierarchical generative model both perform well on CUB Easy, and GANs only show a 0.8732% relative improvement compared to GOGGLES' approach. We observe similar behavior in Table 4 with CUB Moderate. On CUB Hard, where GOGGLES only labelled with $< 88\%$ accuracy, we observe that using GAN-based clustering 13.4732% relative improvement when compared to base GOGGLES. We attribute this improvement to the GAN capturing subtleties in the latent space and being able to model complex distributions, thus making more similar classes more separable. This is a promising for creating more

separability in labelings where the images are quite similar and difficult to distinguish. We found that GANs were slow and often took a while to converge, making them less practical for fast/automatic purposes.

In Table 7 we analyze the performance of various configurations on GOGGLES and the incurred accuracy on the TB-Xray dataset. The improvements with our methods over GOGGLES (if any) is marginal in contrast to large improvements for the CUB dataset. This shows that our methods work better for datasets which are similar to Imagenet, and is limited when taken to other domains such as Xray images. AF48 is more accurate than using all affinity functions. This again shows that we do not need a large number of affinity functions. We also observe that the bottom layers are as informative for clustering as the top layers in this dataset. This makes sense, as the richer features in CUB in the top layers are useful because of domain similarity with Imagenet, but not for Xray images. Finally, the accuracy of the embedding layer trained using the triplet loss does not perform well in our experiments. However, we believe this can be improved with careful design of the loss function and better hyper-parameter tuning, which we leave for future work.

One of the major points we sought out to show with this work was that as the size of the development size increased, we continued to see performance gains for the labelling. In Fig. 4, we observe that GOGGLES performs the same for all samples on TB-Xray, not increasing in accuracy past 76%. We see that with just the vgg-ft method, we increase in accuracy as the size of the development set increases. It is important to note that we used these n samples for both fine-tuning VGG and for the cluster-to-label assignments.

	CUB Easy	**CUB Mod**	**CUB Hard**
GOGGLES	97.9895	91.8592	67.0527
vgg-ft	97.0609	95.9082	79.5637
GAN	98.8263	93.2029	73.9511
vgg-ft + GAN	97.9895	95.2039	79.5322

Table 6: Experimental results over CUB showing the effects of combining affinity function modifications with GAN clustering

	Accuracy
GOGGLES	75.8308
AF48	76.5861
Triplet AF	72.9607
vgg-ft	75.6797
AF48 + vgg-ft	76.5861
GOGGLES + vgg-ft	76.5861
GAN	77.0393
vgg-ft + GAN	76.5861

Table 7: Experimental results over TB-Xray using various techniques

In Table 6 we can see the results of GOGGLES, vgg-ft, GAN, and the combination of both on the three CUB datasets. We chose to only analyze vgg-ft from Table 2 as it was the best result. We notice that the combination does not necessarily improve the accuracy of its component architectures, suggesting that there may be more to the underlying assumptions regarding the interactions between the affinity matrix and the cluster-to-label assignment method.

7 FUTURE WORK

We had initially set out to explore a wide variety of issues with GOGGLES, but we were only able to accomplish a few of these objectives within the scope of the class project; which has been fruitful in and of itself. One of the major things we have not accomplished is exploring multi-class labeling. This was definitely one of the weak points of this research and definitely needs to be fulfilled in the future. We had also hoped to explore more datasets to ensure that the techniques we introduced were invariant to input, but this is left as work to be done outside of this project.

8 CONCLUSION

We set out to look at the weaknesses of GOGGLES and devise new methods for integrating the development set and using more complex methods for clustering the affinity matrix. We found that through utilizing the development set as part of the affinity matrix generation, we are able to get higher labelling accuracy on average in the CUB dataset and when used in combination with other affinity functions in TB-Xray. This work begs many questions and incites further questions regarding improvements to GOGGLES. We focus on two datasets: CUB and TB-Xray, but it would be interesting to see if the results we observe extend beyond these datasets, and how GOGGLES performs on data that is not class uniform, or has embedded noisy data (challenging to label).

REFERENCES

[1] N. Das, S. Chaba, R. Wu, S. Gandhi, D. H. Chau, and X. Chu, "Goggles: Automatic image labeling with affinity coding," *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, May 2020. [Online]. Available: http://dx.doi.org/10.1145/3318464.3380592

[2] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel," *Proceedings of the VLDB Endowment*, vol. 11, no. 3, p. 269–282, Nov 2017. [Online]. Available: http://dx.doi.org/10.14778/3157794.3157797

[3] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3567–3575. [Online]. Available: http://papers.nips.cc/paper/6523-data-programming-creating-large-training-sets-quickly.pdf

[4] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4077–4087. [Online]. Available: http://papers.nips.cc/paper/6996-prototypical-networks-for-few-shot-learning.pdf

[5] S. Qiao, C. Liu, W. Shen, and A. Yuille, "Few-shot image recognition by predicting parameters from activations," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7229–7238.

[6] D. Chen, Y. Chen, Y. Li, F. Mao, Y. He, and H. Xue, "Self-supervised learning for few-shot image classification," *arXiv preprint arXiv:1911.06045*, 2019.

[7] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[8] S. Kim, D. Kim, M. Cho, and S. Kwak, "Proxy anchor loss for deep metric learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[9] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[10] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 1857–1865. [Online]. Available: http://papers.nips.cc/paper/6200-improved-deep-metric-learning-with-multi-class-n-pair-loss-objective.pdf

[12] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "Clustergan : Latent space clustering in generative adversarial networks," 2019.

[13] Y. Yu and W.-J. Zhou, "Mixture of gans for clustering," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 3047–3053. [Online]. Available: https://doi.org/10.24963/ijcai.2018/423

[14] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep., 2011.

[15] S. Jaeger, A. Karargyris, S. Candemir, L. Folio, J. Siegelman, F. Callaghan, Z. Xue, K. Palaniappan, R. K. Singh, S. Antani *et al.*, "Automatic tuberculosis screening using chest radiographs," *IEEE transactions on medical imaging*, vol. 33, no. 2, pp. 233–245, 2013.

[16] S. Puch, I. Sánchez, and M. Rowe, "Few-shot learning with deep triplet networks for brain imaging modality recognition," 2019.

[17] M. Kaya and H. Bilge, "Deep metric learning: A survey," *Symmetry*, vol. 11, p. 1066, 08 2019.

[18] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, vol. 32, pp. 323 – 332, 2012, selected Papers from IJCNN 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457

[19] W. Louhichi, "Automated surface finish inspection using convolutional neural networks," Ph.D. dissertation, Georgia Institute of Technology, 2019.

[20] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan *et al.*, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.

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

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

[23] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.