CSE 527 PROJECT REPORT

Adversarial Feature Hallucination Networks for Few-Shot Learning

Srinivas Kandari - 112713946
Sandeep Acharya- 112715935

1. Objective:

The objective of this project is to implement a novel approach in the few shot learning paradigm; Adversarial Feature Hallucination Networks for Few-Shot Learning. This uses conditional WGAN, together with two novel regularizers for improving the discriminability and diversity of the generated samples.

2. Background:

2.1 Few Shot Learning problem:
Few Shot learning is a machine learning problem that addresses how to accomplish the learning in scenarios where data is scarce.For Instance, consider Office Face Recognition, You want your algorithm to classify the person's face while having only one or few images of that person in your database(Training data).
Conventional classification models fail to achieve this as their accuracy is data intensive, Provided that we have very few labeled images for each class it is extremely difficult to achieve a sizable accuracy with conventional models.

There are three classes of algorithms that provide solution for this problem

2.1.1 Transforming the existing data:
Here the samples for each class are transformed to a better representation such that the samples in this representation have better discriminability, which improves the quality of data and aids in learning. Even a simple linear model can achieve satisfactory results with this.

2.1.2Meta-Learning:
Instead of learning a classifier that learns the distinctions of each class we learn a model that measures similarity of two images. Given two images the model Ideally gives very less score if they don't belong to the same class and it gives a very high score when the two images are actually from the same class. Here the model; instead of learning and remembering what features a particular class constitutes; it learns to extract the features that decide the class. It learns to learn. Hence, this paradigm is called Meta-Learning.
2.1.3 Data Augmentation:
Here, the issue of having less data is addressed by synthesising more data with the help of existing data. For this approach to work it is pivotal that the generated data is

discriminable and diverse while maintaining good intra-class variance. Many existing approaches fail to achieve these three resulting in undesirable results.

This project Falls under the data augmentation category. It addresses the issues in Data Augmentation by using two novel regularizes as follows:

2.2 Intra class Variance:
To achieve precise decision boundaries it is important that there is sufficient variance among the samples in a class. A few samples have limited intra-class variance, Therefore we are encouraged to add more data and this approach makes sure that the new samples have new features by conditioning them on existing features of that class and ensures intra-class variance.

2.3 Discriminability:
It is important that the new data produced is discriminable. Having data that is not discriminable would drastically damage the learning process. Here, we use a regularizer that compares the generated features with the existing features of all the classes in the data. And makes sure that the generated features have good correlation with the features of its class and less correlation with features of other classes. Thus ensures Discriminability.

2.4 Diversity:
GANs usually face a problem called mode collapse where the GAN ignores the input noise vector and gives feature vectors of very few classes. Here, we use a **Anti-mode collapse regularizer** that makes sure two generated feature representations are dissimilar as to the factor of dissimilarity between their input noise vectors.

3. Approach:

3.1 Wasserstein GAN :
Vanilla GAN is highly unstable in training. Martin Arjovsky, Soumith Chintala, and Leon Bottou have analyzed the convergence properties of the objective function of GAN. They proposed Wasserstein GAN (WGAN) which uses Wasserstein distance and is shown to have better theoretical properties. The following variant of WGAN will be used. This variant optimized the following min-max problem.

$$\min_G \max_D \underset{\tilde{\mathbf{x}}\sim\mathbb{P}_g}{\mathbb{E}}[D(\tilde{\mathbf{x}})] - \underset{\mathbf{x}\sim\mathbb{P}_r}{\mathbb{E}}[D(\mathbf{x})] + \lambda \underset{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}{\mathbb{E}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2],$$

The first two terms approximate the Wasserstein distance and the third term penalizes the gradient norm of x.

3.2 Adversarial Feature Hallucination Networks :
First, Few Shot Learning (FSL) can be defined as given a distribution of tasks P(T), a sample task T ~ P(T) is a tuple T = (S_T, Q_T) where the S_T Is a support set and Q_T is a query set. If there are K labeled samples from each of the N classes, then this is known as K-shot N-way classification. The objective is to minimize the classification prediction risk of Q_T according to S_T.

AFHN does this by using a general conditional WGAN based FSL framework and 2 regularization terms.

3.3 FSL framework with conditional WGAN :

s = F(x) , where (x,y) ∈ S_T

For, k shot classification, we just average the feature vectors.

We first generate two noise variables z_1 and z_2 ~ N (0, 1)

$$\tilde{\mathbf{s}}_i = G(\mathbf{s}, \mathbf{z}_i),\ i = 1, 2.$$

The generator G aims to get the above result as similar as s. The discriminator D, taking z_i and s as input, tries to identify the above result as false and s as real.

Following is the adversarial training objective within the WGAN framework.

$$L_{GAN_i} = \underset{(\mathbf{x},y)\sim S_\mathcal{T}}{\mathbb{E}}[D(\tilde{\mathbf{s}}_i, \mathbf{z}_i)] - \underset{(\mathbf{x},y)\sim S_\mathcal{T}}{\mathbb{E}}[D(\mathbf{s}, \mathbf{z}_i)] + \lambda \underset{(\mathbf{x},y)\sim S_\mathcal{T}}{\mathbb{E}}[(\|\nabla_{\hat{\mathbf{s}}_i} D(\hat{\mathbf{s}}_i, \mathbf{z}_i)\|_2 - 1)^2],\ \ i = 1, 2.$$

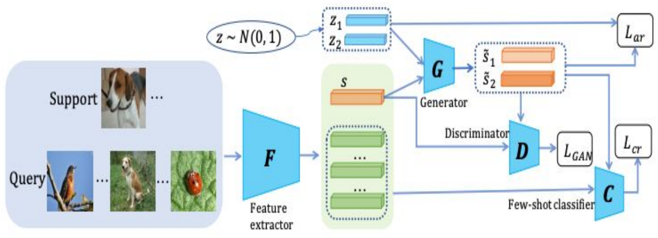

Figure 1: AFHN framework. Input - A support set and a query set. Each image in the support set is fed to Feature Extractor F which gives feature embedding s. Given s, Generator G produces 2 fake features by combining s with 2 randomly sampled variables z_1 and z_2. Discriminator D discriminates between real (s) and fake features s_1 and s_2. The proposed Few shot classifier classifies the features of the query images based on fake features s_1 and s_2. L_{cr} is the resultant classification loss.

3.4 Classification Regularizer :

Given an image X_q from Query set the probability of it belonging to a class 'y' is given by the following equation, where the feature representation of X_q is given by 'q'. Here we measure the similarity between q and s^{-}_{i}, the synthesized feature for class y. And normalizes it with sum of similarity score of all other classes j

$$P(y_q = y|\mathbf{x}_q) = \frac{\exp(\cos(\tilde{\mathbf{s}}_i, \mathbf{q}))}{\sum_{j=1}^{N} \exp(\cos(\tilde{\mathbf{s}}_i^j, \mathbf{q}))},$$

The following is the loss function that optimizes the labelling the function.

$$L_{cr_i} = \underset{(\mathbf{x}_q,y_q)\sim Q_\mathcal{T}}{\mathbb{E}}\left[\frac{1}{N}\sum_{y=1}^{N} y \log[-P(y_q = y|\mathbf{x}_q)]\right],$$

3.5 Anti-Collapse Regularizer :

This regularizer makes sure that the synthesised feature vector depends on the input noise vector 'z'. The variance in the output feature vector s^{-}_1, s^{-}_2 should be proportional to the variance of input noise vectors z_1, z_2. This ensures that the synthesized features doesn't suffer mode collapse

$$\mathcal{L}_{ar} = \underset{(\mathbf{x},y)\sim S_\mathcal{T}}{\mathbb{E}}\left[\frac{1 - \cos(\tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2)}{1 - \cos(\mathbf{z}_1, \mathbf{z}_2)}\right].$$

3.6 Final Training Objective :

Combining the three loss components the final loss function is

$$\min_G \max_D \sum_{i=1}^{2} L_{GAN_i} + \alpha \sum_{i=1}^{2} L_{cr_i} + \beta \frac{1}{L_{ar}},$$

Where, alpha and beta are hyper-parameters.

3.7 Algorithm :

Algorithm 1. Proposed FSL algorithm

```
Input: Training set $\mathcal{D}_t = \{\mathcal{X}_t, \mathcal{Y}_t\}$, parameters $\lambda$, $\alpha$, and $\beta$.
Output: Feature extractor $F$, generator $G$, discriminator $D$.
1. Train $F$ as a standard classification task using $\mathcal{D}_t$.
while not done do
  // Fix G and update D.
  2. Sample from $\mathcal{D}_t$ a batch of FSL tasks $\mathcal{T}_i^d \sim p(\mathcal{D}_t)$.
  For each $\mathcal{T}_i^d$ do
    3. Sample a support set $S_\mathcal{T} = \{\{\mathbf{x}_{i,j}\}_{i=1}^K, y_j\}_{j=1}^N$ and
       query set $Q_\mathcal{T} = \{\{\mathbf{x}_{k,j}\}_{k=1}^Q, y_j\}_{j=1}^N$.
    4. Compute prototypes of the $N$ classes $\mathcal{P} = \{\mathbf{s}_j\}_{j=1}^N$,
       where $\mathbf{s}_j = \frac{1}{K}\sum_{i=1}^K F(\mathbf{x}_{i,j})$.
    5. Sample $N$ noise variables $\mathcal{Z}_1 = \{\mathbf{z}_1^j\}_{j=1}^N$ and
       variables $\mathcal{Z}_2 = \{\mathbf{z}_2^j\}_{j=1}^N$.
    6. Generate fake feature sets $\tilde{\mathcal{Z}}_1 = \{\tilde{\mathbf{z}}_1^j\}_{j=1}^N$
       and $\tilde{\mathcal{Z}}_2 = \{\tilde{\mathbf{z}}_2^j\}_{j=1}^N$ according to Eq. (3).
    7. Update $D$ by maximizing Eq. (8).
  end For
  // Fix D and update G.
  8. Sample from $\mathcal{D}_t$ a batch of FSL tasks $\mathcal{T}_i^g \sim p(\mathcal{D}_t)$.
  For each $\mathcal{T}_i^g$ do
    9. Execute steps 3 - 7.
    10. Update $G$ by minimizing Eq. (8).
  end For
end while
```

4. Implementation

4.1 Dataset:

We used the CIFAR-100 dataset to train and evaluate our model. CIFAR-100 contains 100 classes with 500 samples of 32×32 color images per class in the training set. Test set contains 100 samples for each class. These 100 classes are divided into 80 and 20 classes respectively for sampling tasks for meta-training and meta-test.

4.2 Data Loaders :

One of the most challenging parts of the implementation was to provide data for training and testing in the format of a FSL task. We used 80 classes of the training data for training GAN. The data was provided as a batch of FSL tasks, where each task contains a support set and query set. Support set contained N classes with K randomly chosen examples for each class. Query set contained 1 randomly selected example for each of the N classes. For each FSL task, we made sure that the query set and support set didn't overlap.

Each FSL task for testing also contains a support set and a query set. Support set was generated from the N classes out of the remaining 20 unseen classes of the training set. Query set was generated from the test dataset. Support set was used to augment new features, thus increasing the size of the support set. Support set was used to train a softmax classifier with N output classes. Then, we predicted the query set using this classifier.

4.3 Feature Extraction :

We used a pre-trained resnet18 model for feature extraction. We further trained the model using our training set (of 80 classes). Then, the final classifying layer was removed. This was used to extract features from the training set as well as the test set.

4.4 CWGAN-GP:

We implemented a CWGAN-GP that uses wasserstein distance as the loss metric along with gradient penalty term that ensures lipschitzness to be close to 1. This is a conditional GAN, i.e, we pass the conditional context of the class from which we want to sample, these contexts are means of the feature vectors of each class features from support set, we train the generator to generator to generate samples that are realistic (Adversarial loss), Discriminable (Classification regularizer) and has intra-class variance (Anti-collapse regularizer).

Note: Here GAN is trying to sample feature representation of the images(Resnet18 representation) and not pixel images.

4.5 Architecture:

The generator Network has two linear neural layers with 1024 and 512 neurons respectively and the first layer activation is LeakyReLU while the second layer uses ReLu activation.

The Discriminator has two layers; The first layer has 1024 neurons with LeakyReLU activation while the second layer has one neuron. Note that we don't use sigmoid at the end (vanilla GANs) as the WGANs range is not 0-1.

For, every 5 batches of Discriminator training we train Generator once.
By the end of this training we would have learnt a generative model that maps class context input to Discriminable class samples.

4.6Tuning:
We have tried training the GAN for different numbers of epochs and found 100 to be effective. For, the number of times Discriminator should be trained for every one training step of Generator, 5 was found to be effective. In the paper though they use WGAN, they have inserted a sigmoid layer at the end of the discriminator, However, WGANs main idea is to not limit the output of discriminator to 0-1. Upon experimenting we found not putting a sigmoid to be more stable than with sigmoid.

4.7 Testing:
During the test stage, FSL task T' = (S_T' , Q_T'). We first augment S_T' with the learned generator G. Then, we train a classifier with the augmented support set. This classifier classifies samples from Q_T'.

$$\min_{\theta} \mathop{\mathbb{E}}_{(\mathbf{s},y)\sim\hat{S}'_{\mathcal{R}}} \log[-P(y|\mathbf{s};\theta)],$$

This is a standard Softmax classifier f_c. Theta is the parameter of f_c. S'$_R$ is the augmented support set.

Note: We build a new classifier for each FSL task and we have ensured that the classes used in FSL tasks don't overlap with those used to train the Generator.

The classifier is a Fully connected neural network with three layers, containing 1024, 512 and N (N-class FSL) neurons respectively. The first two layers use Tanh activations with dropout, Batch normalization has been applied on the output of the second layer. Finally, the outputs have been mapped to probabilities using softmax.

5. Results:
5.1 Generated Images:
 We have generated images using context formed from 5 images per class (5 shot). And projected the synthetically generated features to 2D using t-SNE and the result is shown in Fig 3.

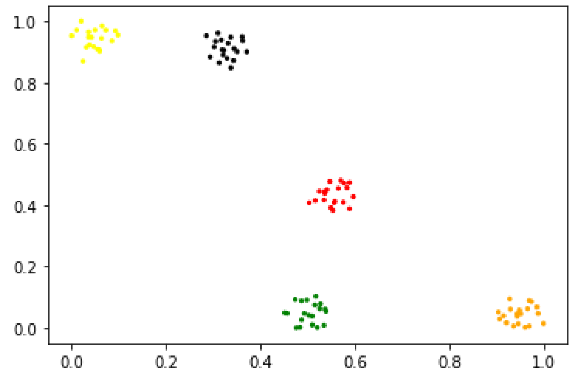

Fig 3. visualization of synthesized feature embeddings

5.2 Accuracy on FSL Tasks:
We have sampled FSL tasks as 5 way 5 shot and 5 way 1 shot. And we have also experimented with adding different numbers of synthesized examples for these FSL tasks the accuracies are depicted in the figure. The best Accuracy for One shot learning was 57% when we added 100

synthesized examples, and it was 69% for 5 shot learning tasks.

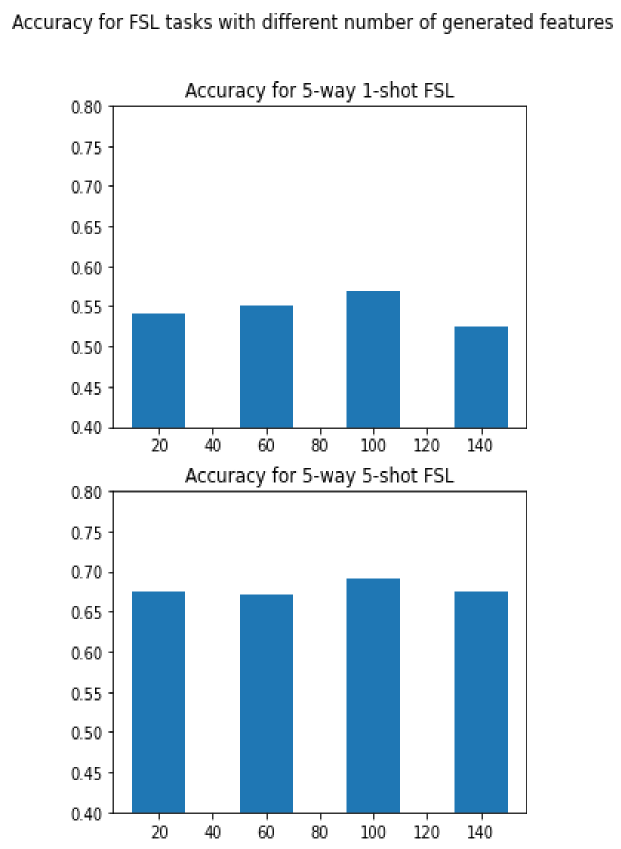

Fig 2: Accuracy for FSL tasks with different number of generated features

Contribution :

Implementation of this paper doesn't exist anywhere online so we implemented everything from scratch. However, some tutorials, videos, articles and other papers were referred.
Project contribution among team members is around 50-50% each.

REFERENCES :

[1] Martin Arjovsky, Soumith Chintala, and Leon Bottou. ´ Wasserstein generative adversarial networks. In ICML, 2017.
[2]openaccess.thecvf.com/content_CVPR_2020/papers/Li_Adversarial_Feature_Hallucination_Networks_for_Few-Shot_Learning_CVPR_2020_paper.pdf
[3]
https://github.com/yaoyao-liu/mini-imagenet-tools