Summary of “How to train your pre-trained GAN models”

Abstract/Introduction - Summary

GAN problems:

-Take a lot of data and computational resources

-Unstable training; GAN may not always converge

The attempted solution to those problems is transfer learning

But issue of mode collapse still exists!

This paper goes over the "latest transfer learning methods as a solution to the problem". StyleGAN model is used as the GAN model in this paper.

Paper’s result: model did not overfit, plus model "outperformed existing methods" at a bunch of datasets used in the experiments

Methods

Various methods of transfer learning, which are:

- Fine-tuning
- Freezing
- Scale/shift
- Generative Latent Optimization (GLO)
- MineGan
- L2-Starting Point (SP)
- Feature Distillation (FD)

Method: Fine-tuning

-Initialize new target model using pre-trained weights

-Then, retrain weights of all layers: keep training model on top of pre-trained weights. But use a low learning rate to avoid drastically changing the old weights (want to keep the model's old knowledge to some extent)

-Can be expensive time-wise compared to freezing layers

-Needs 'regularization' to avoid overfitting

Method: Freezing

-‘Freeze’ the weights of some layers so that they remain unchanged during training. Only non-frozen weights will be trained.

-A possible implementation of freezing: change only variables in the output layer while ignoring variables of other layer. As a result, every non-output-layer weight stays the same. Only the weights of the output layer are trained.

Method: Scale/Shift

-This method “updates the normalization layer only when the weights are frozen” [1]

-uses pre-trained network’s knowledge “to learn small datasets in other domains” [1]

-involves updating parameters during supervised learning. "This method focuses on batch statistics of the network hidden layer, scale, and shift parameters." [1]

Method: Generative Latent Optimization (GLO)

-Defines loss = L2 loss + perceptual ("perceptual loss is defined using the feature reconstruction loss and the style reconstruction loss") and "fine-tunes the generator with a supervised learning method". Optimizes "generator and latent code together to avoid overfitting"

-During model training, use a "latent vector z" which "tracks the correspondence between the sample and its representatives".

-Overall: utilizes feature reconstruction loss and style reconstruction loss, optimizes generator. No discriminator used in GLO.

Method: MineGAN

-Involves mining the 'most meaningful knowledge' from pretrained GAN(s)

-Good for avoiding mode collapse and training instability

- "can be difficult to generalize when the source and target distributions do not share support" --> not as effective when source and target models unsimilar

Method: L2-Starting Point (SP)

-Purpose is to "keep features learned from the source dataset well" (in other words, making sure model doesn't go too far from the pre-trained weights)

-Compared to a typical L2 penalty, L2-SP adds a penalty based on the starting point (such as a starting point of pre-defined weights)

-"L2-SP is easier to implement than freezing the initial layer in the network"

-Results of trying "L2-SP as a generator, a discriminator, and both" were not good, but there was possible room for improvement: "since freezing layers can be seen as providing an infinite weight of L2-SP for the selected layer and weight of ‘0’ for the unselected layer, it is considered that the appropriate weight for each layer can perform better."

Method: Feature Distillation (FD)

-Transfer features from large & deep model to small & shallow model. Involves large model synthesizing a "transfer dataset" that outputs a soft target, which will be used in training of the small model

-Small model is "trained using the transfer dataset and the training dataset"

-Small model will have both the soft target and the hard target

-Soft target = the probability of each class (probability determined by softmax output)

-Hard target = the label value (the class itself as determined from the class probability)

In practice, soft target can be "adopted as a realistic target" when training a new model; that means it may effectively be a hard target to the small model

-Train by calculating loss of both targets

Experiment Set-up

-Hardware limitations --> could not generate images "with a resolution higher than 256 x 256"

-Software used: Compute Unified Device Architecture (CUDA), Torch, and python

-Model used: StyleGAN, pretrained on Flickr Faces High Quality (FFHQ) dataset

-"number of layers that can be frozen in the generator and the discriminator is the same. Thus, an accurate comparative analysis is possible."

Datasets

-6 datasets used. “Stanford Cars, Stanford Dogs, Oxford Flower, Caltech-256, CUB-200–2011, and Insect-30” [1] Sizes respectively were 196 classes & 8144 images; 120 classes & 20580 images; 102 classes & 8189 images; 256 classes & 30609 images; 200 classes & 11788 images; 30 classes & 28896 images

-Screen Scraping was used for one of the datasets

-Crop only the relevant part of the image, keep that as new image

-Image size used in training: 256 x 256

-Iteration = 50000

Implementation

-When “injecting labels into the discriminator”, the loss function used logistics; the activation function used softplus (ReLU variant)

-When using L2-SP or FD methods: they define loss as existing loss + the Mean Squared Error (MSE). Used ADAM optimizer.

Specific numbers: mini-batch = 8, image size = 256, number of samples for evaluation = 5000, number of samples used for each training phase = 50,000, basic step size = 6, step size for evaluation = 1,000, step size for model save = 10,000

Evaluation metrics

-Inception Score (IS) and the Fréchet Inception Distance (FID)

-Existing evaluation metrics: Precision, recall, F1 score, coverage, density

-In this paper, they used FID, coverage, and density

Results - Overview

-Results shown include images generated by:

1) base StyleGAN model

2) partial fine-tuning method

3) Freeze G method

Results - Overview

-Compared various methods (fine-tuning, freezing) over different datasets; “fine-tuning is better able to adapt to the specific properties of the target dataset than the freezing method.”

Results

Partial fine-tuning method had the best performance in the ‘Stanford Cars’ dataset.

Freeze method had the best performance in the ‘Stanford Dogs’ dataset

"As a result of the analysis, we found that each item represented a similar latent code and that similar latent codes shared similar meanings even after freezing"

Results

-Over two datasets (Stanford Cars and Stanford Dogs), the paper compared freezing vs partial-fine tuning methods.

"For the Stanford Cars dataset, Freeze D fixed up to discriminator layer 5, and Freeze G fixed it up to generator layer 3. Partial fine-tuning method was fixed up to generator layer 4 and discriminator layer 4. For the Stanford Dogs dataset, Freeze D fixed up to discriminator layer 5, and Freeze G fixed it up to generator layer 2. Partial fine-tuning method was fixed up to generator layer 5 and discriminator layer 7." [1]

Results - Evaluating by FID

In Stanford Cars dataset: best method = partial fine-tuning method (FID 10.84). worst method = MineGAN with GLO (FID 33.65)

In Stanford Dogs dataset: best method = Freeze G (FID 29.83). worst method = scale/shift (FID 57.30)

Note for scale/shift: they acknowledge possibility of using another layer instead of the current normalization layer which may improve performance

Also acknowledging hyperparameters such as learning rate which may affect performance degradation (of those methods that performed badly)

Results - Evaluation by coverage and density

When evaluating by coverage/density, they slightly changed the layer numbers in Freeze D and Freeze G

In *Stanford Cars*:

-by coverage: best = partial-fine tuning, worst = MineGAN

-by density: best = fine-tuning with GLO, MineGAN = worst

In *Stanford Dogs*:

-by coverage: best = FD, worst = scale/shift

-by density: best = fine-tuning with GLO, worst = L2-SP

showed graphs of FID, Coverage, Density values over epochs of existing methods (already mentioned in paper, such as fine-tuning with GLO or MineGAN with GLO) and paper's proposed method

Results

Next tried different datasets (the ones with flowers) which they said was harder to learn because of "large shift in distribution during training"

Oxford Flower dataset:
-looking at coverage, Freeze D and partial fine-tuning were best, while freeze G was worst
-explanation: fine tuning > freezing because this new dataset is more different than the previously learned dataset

CUB-200–2011 dataset:
-looking at coverage: freeze D best, freeze G worst.
-looking at density, Freeze G best, partial fine-tuning worst

Caltech-256 dataset:
-looking at coverage: freeze G best, freeze D worst
-looking at density: partial fine-tuning best, and fine-tuning worst

-Showed performance at epochs for freeze D, freeze G, partial fine-tuning, fine-tuning

Results

Insect dataset:

-mostly talked about which methods synthesized which insect species the best, like "Freeze D best synthesized the 8 species" (list of species here). Also looked at coverage and density values to see which method synthesized which species better.

Conclusion

Conclusion: "The proposed method divides the generator and discriminator, and then freezing or partially fine-tuning. Based on the FID value alone, the proposed method was superior to the existing method"

-Proposed method was the best method

-Only a partial solution to GAN instability problem

Reference

Park, S. W., Kim, J. Y., Park, J., Jung, S. H., & Sim, C. B. (2023). How to train your pre-trained GAN models. *Applied Intelligence, 53*(22), 27001-27026.