WEIHE WANG

Machine Learning Capstone Project Proposal

Topic: Generative Adversarial Network for CIFAR-100 dataset

Prepared by: Weihe Wang

May 18, 2018

WEIHE WANG

PROPOSAL

Objective

Build a generative adversarial neural network (GAN) and trained it with CIFAR-100 dataset.

Domain Background

CIFAR-100 is a classic dataset consists of 60,000 color images of size 32*32. Numerous research take CIFAR-100 test result as a benchmark performance metric for their models. GAN is a class of deep neural networks that gained tremendous interest due to its capability to learn patterns and create new data. We have seen GANs creating paintings by training them with masterpieces datasets.

Problem Statement

Image classification of 100 classes is itself a challenging problem. The goal of the project is to train a generative adversarial neural network capable of generating clear images similar to those in the CIFAR-100 dataset. The second section of the project uses semi-supervised learning to train a classifier with two sources of data: real images and fake images from the generator.

Datasets and Inputs

The CIFAR-100 dataset consists of 100 classes of 600 images in each class. Also each image is labeled with one fine-class label (100 in total) and one super-class label (20 in total). 10,000 images are set aside as test set. We intend to use the CIFAR-100 dataset in Keras for the purpose of the project.

Solution Statement

We propose to use DCGAN architecture for building the neural networks. For the task of generating clear images, the architecture of the generator consists of transpose convolutional layers, batch-norm layers and output layer. Details are specified in Project Design section.

Benchmark Models

ELU-based CNN archives 24.28% test error rate on CIFAR-100 dataset, which is lower than classical CNN such as AlexNet.

Evaluation Metrics

Generator loss is usually not a very good indicator of performance for generator evaluation. During training, we will generate samples intermittently to observe the generator performance. For semi-supervised learning, we will evaluate the performance of the discriminator with classification accuracy.

Project Design

The Python packages we will be using include Numpy, Keras, Matplotlib and Tensorflow.

Before training any machine learning models, we should consider data quality and completeness. For our proposed project, we are fortunate to have high quality dataset with labels from Keras.dataset.

The first part of the project aims to train a GAN that can generate images in similar categories specified in CIFAR-100 dataset. Specifically, we expect the performance of the generator good enough to generate images that can be recognized as belonging to one of the 20 superclasses. All training images are used as real images. The network architecture for the generator is specified below. Each convolutional layer follows a batch-normalization layer.

We design the discriminator network to be a regular deep convolutional neural network and trained with real images as well as fake images from generator.

For the second section of the project, we train a GAN using semi-supervised learning. The discriminator is trained with two sources of data: real images and fake images from generator network. The architecture of the generator is similar to that in part one.

For the discriminator architecture, we include a *feature matching layer*, which is equivalent to training the generator to learn from discriminative features instead of just the outputs.

There are two sources of loss for the discriminator. First is the loss when it tries to differentiate between fake and real, and second comes with classification. Therefore we have the following formula for discriminator loss.

$$d_{loss} = d_{lossReal} + d_{lossFake} + d_{lossClassification}$$

All the losses can be calculated using discriminator logits and class logits with categorical cross entropy function.

The optimizer of choice is Adam optimizer in Tensorflow.

Network Architecture for the Basic DCGAN Generator

Layer	Output Size	Number of Filters	Filter Size	Activation
Input Layer	100	NA	NA	NA
Fully Connected Layer	4 * 4 * 512	NA	NA	None
Convolution Layer 1	8 * 8 * 256	256	5 * 5	Leaky ReLu
Convolution Layer 2	16 * 16 * 128	128	5 * 5	Leaky ReLu
Convolution Layer 3	32 * 32 * 64	64	5 * 5	Leaky ReLu
Output Layer	32 * 32 * 3	3	5 * 5	Tanh

This completes the project design. We will see how the actual project built up and adjust the procedures accordingly.