VISUAL RECOGNITION ASSIGNMENT 3

...

CONVOLUTIONAL NEURAL NETWORK (CNN), ALEXNET AND YOLO

IMT2019003 Aditya Vardhan · IMT2019028 Divyam Agrawal · IMT2019031 Gagan Agarwal

1 Play with CNNs

1.1 Introduction

We built a CNN model made up of 2 convolutional layers and 3 fully connected layers for image classification on CIFAR-10 dataset. We have applied ReLU, tanh and sigmoid activation functions with and without batch normalization. We have also tried optimizers With and without momentum and adaptive learning rates.

1.2 CIFAR 10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The 10 classes in the dataset are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.

1.3 What is CNN?

CNN is a deep learning algorithm that is frequently used to analyse visual imagery. Unlike traditional methods, where image features are hard engineered, CNN allows us to give input as the image is and let the image learn the features/characteristics on its own.

CNNs are made up of numerous layers of artificial neurons. The output of one layer of CNN is passed into the next. The first layer of CNN typically recognises fundamental properties such as horizontal or vertical edges. As you progress further into the layer, it begins to recognise higher-level features such as objects and faces. CNN layers can be convolutional, pooling, or fully-connected.

1.4 Activation functions

We used the following activation functions:

1.4.1 ReLU

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. So it can be written as $f(x) = max(0, x)$.

The ReLU function is non-linear around 0, but the slope is always either 0 (for negative values) or 1 (for positive values).

Rectified linear units, compared to sigmoid function or similar activation functions, allow faster and effective training of deep neural architectures on large and complex datasets.

1.4.2 Sigmoid

When using a traditional CNN for image classification, the output layer comprises N neurons, where N is the number of image classes to classify. We want each output neuron to indicate the probability that we saw each image class.The sigmoid function is useful for expressing probability as it's domain consists entirely of real numbers, yet its range is 0 to 1.

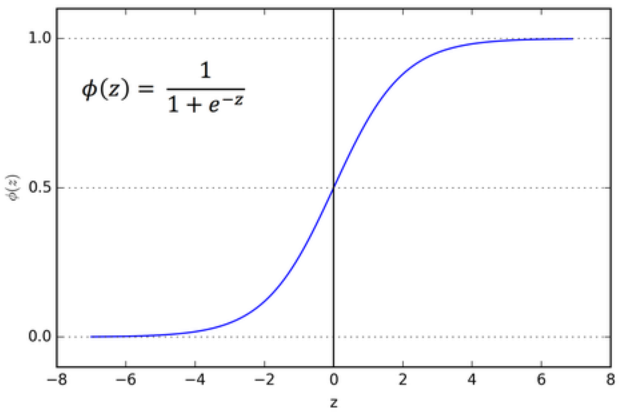

Figure 1: Sigmoid Function

1.4.3 Tanh

The tanh function is simialr to logistic sigmoid but better. The range of the tanh function is $-1 \rightarrow 1$. tanh is also sigmoidal (s - shaped). The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph which helps with quicker convergence to the optimal solution.

1.5 Batch Normalization

Batch normalization, is a process to make neural networks faster and more stable through adding extra layers in a deep neural network. The new layer performs the standardizing and normalizing operations on the input of a layer coming from a previous layer. The normalizing process takes place in batches, not as a single input.

1.6 Results

Pre-processing: Transformed data to tensor and normalised with mean = (0.5, 0.5, 0.5) and sd = (0.5, 0.5, 0.5).

Hyper-parameters:

- Batch size: 8

- Learning rate : 0.001
- Epochs : 10
- Loss function : Cross-entropy loss
- 2d Max pooling

Optimizer: Adam

Model	Batch Normalization	Time(s)	Accuracy
RelU	Yes	524.224	72.68
Sigmoid	Yes	517.322	70.13
Tanh	Yes	520.06	50.68
RelU	No	480.986	70.86
Sigmoid	No	483.894	9.94
Tanh	No	482.65	46.76

Optimizer: SGD with momentum = 0

Model	Batch Normalization	Time(s)	Accuracy
RelU	Yes	425.103	74.48
Sigmoid	Yes	423.998	74.89
Tanh	Yes	425.398	56.99
RelU	No	404.055	72.87
Sigmoid	No	401.325	10.13
Tanh	No	402.035	57.09

Optimizer: SGD with momentum = 0.9

Model	Batch Normalization	Time(s)	Accuracy
RelU	Yes	451.042	74.22
Sigmoid	Yes	451.312	74.51
Tanh	Yes	454.316	53.04
RelU	No	429.723	67.84
Sigmoid	No	428.659	10.02
Tanh	No	426.997	54.95

1.7 Observations

- We got the top most accuracy of 74.89% with Sigmoid activation function on Batch normalised data along with SGD optimizer with no momentum.
- On applying Batch Normalisation, the time taken to train the network increases drastically.
- On introducing momentum the training time increased but it didn't change the accuracy much.
- Adaptive learning rate helped us increase the accuracy in non batch normalized data but it reduced accuracy on batch normalized data.

2 CNN as a feature extractor

2.1 Introduction

We used the AlexNet model as a feature extractor on the STL-10 dataset and applied models such as Logistic regression, Linear SVC and Gaussian Naive bayes. After that, we also applied the same model on Bike vs Horses datatset.

2.2 STL10 dataset

We have used the STL-10 dataset. The STL-10 dataset is an image recognition dataset for developing unsupervised feature learning, deep learning, self-taught learning algorithms. It is inspired by the CIFAR-10 dataset but with some modifications. In particular, each class has fewer labelled training examples than in CIFAR-10, but a very large set of unlabelled examples is provided to learn image models prior to supervised training.

The 10 classes in the dataset are airplane, bird, car, cat, deer,dog, horse, monkey, ship, truck.5000 images are in the train set and 8000 images are in the test set

2.3 What is AlexNet?

AlexNet is a CNN architecture which consists of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and 1 softmax layer. Each convolutional layer consists of convolutional filters and a nonlinear activation function ReLU. The pooling layers are used to perform max pooling. Input size is fixed due to the presence of fully connected layers.

AlexNet is divided into 3 Sequential Parts, features, avgpool and classifier.

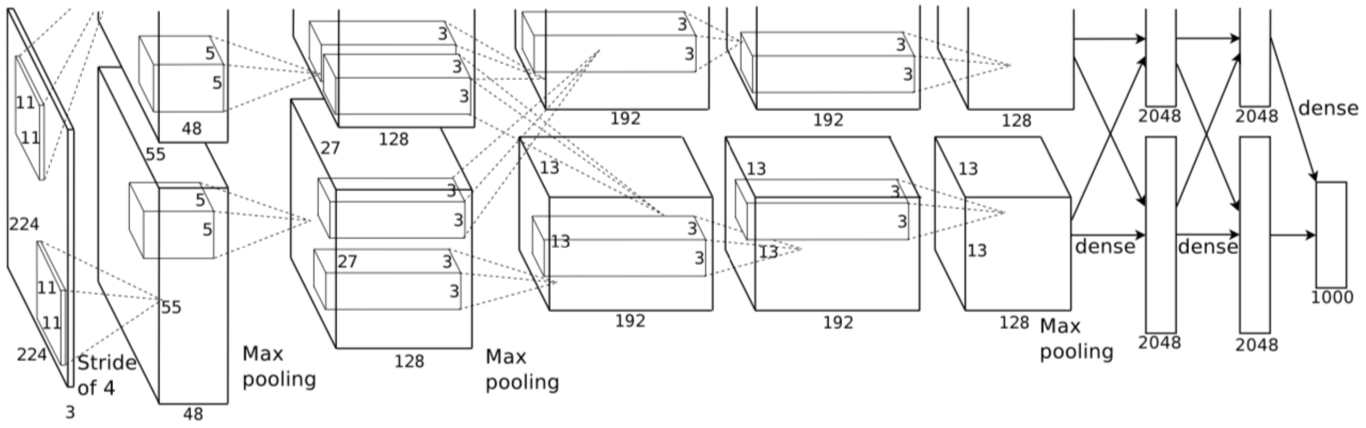

Figure 2: AlexNet Architecture

2.4 Results

Pre-processing: Resized each image to 256 x 256 then centre cropped to 224 x 224. Converted data to tensor and normalised with mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
We have used pretrained AlexNet as a feature extractor and then applied other models on top of it.

2.4.1 STL-10 Dataset

Model	Accuracy
Logistic regression	88.4%
Linear SVC	87.4%
Gaussian Naive Bayes	77.8%

2.4.2 Bike vs Horses Dataset

Model	Accuracy
Logistic regression	100%
Linear SVC	100%
Gaussian Naive Bayes	100%

2.5 Conclusion

Logistic Regression gave the best results with AlexNet as feature extractor for STL 10 dataset. For Bike vs Horses each model gave an accuracy of 100%.

3 Auto Detection

3.1 What is YOLO?

YOLO(You Only Look Once) is an algorithm that uses Convolutional Neural Networks (CNN), to detect objects in images. The algorithm stands true to its name in the sense that only one forward propagation through a neural network is needed for object detection. The YOLO algorithm has been implemented and open sourced. Till now 5 versions of YOLO implementations have been released. In this assignment, we've used the latest version, YOLO-v5.

3.2 How does YOLO work?

1. The image is divided into several equal-sized grids. Objects that appear within grid cells will be detected by each grid cell.
2. YOLO predicts the height, width, centre, and class of objects using a single bounding box regression. A bounding box is an outline that draws attention to a certain object in a photograph. Each bounding box in the image has width, height, class, bounding box centre and confidence as its 5 attributes.
3. YOLO uses IoU(Intersection Over Union) to create an output box that properly surrounds the objects being detected.
    (a) The concept of intersection over union (IoU) illustrates how boxes overlap in object detection.
    (b) The bounding boxes and their confidence scores are predicted by each grid cell. If the anticipated and real bounding boxes are identical, the IoU is 1. This approach removes bounding boxes that aren't the same size as the actual box.

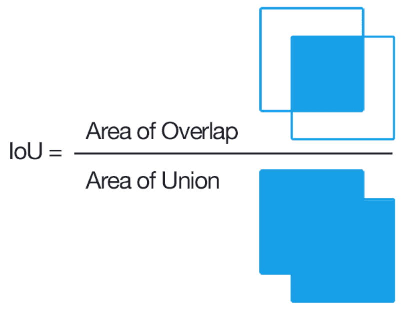

Figure 3: IoU formula

3.3 Where does the method(YOLO) work well?

1. YOLO is exceptionally fast since it predicts multiple bounding boxes and class probabilities for those boxes using a single neural network. So, it works well for real time detection.
2. YOLO encodes contextual information about classes as well as their appearance because it views the complete image throughout training and test time. Because it can't see the greater context, the top detection method Fast R-CNN misidentifies background patches in an image as objects. When compared to Fast R-CNN, YOLO produces less than half as many background mistakes.
3. When applied to new domains or unexpected inputs, YOLO is less likely to break down due to its strong generalizability.

3.4 Where does the method(YOLO) fail?

1. When compared to Faster R CNN, it has a lower recall and a higher localization error.
2. Small items are difficult to detect.
3. Since each grid can only suggest two bounding boxes, it has difficulty detecting nearby items.

3.5 Output / Observations:

1. Metrics:
   (a) Precision = 0.949
   (b) Recall = 0.86
   (c) Mean Average Precision or mAP (for IoU > 0.5) = 0.937
   (d) Average of mAP(s) for IoU ranging between 0.5 to 0.95 = 0.705
2. Above metrics were observed for the following values of hyperparameters:
   (a) Image size = 416
   (b) Batch size = 8
   (c) Number of epochs = 150
3. We've used the YOLOv5s of the YOLO-v5 release, as it was fastest and gave an acceptable amount of accuracy.

4. Output for auto(s) detection for the test images: