NANYANG TECHNOLOGICAL UNIVERSITY

CZ4042: Group Project

Name	Matric No.
Kee Kai Teng	U2020909E
Zhong Shaojie	U2020758K
Darryl Chia Shen Yang	U2022120B

Introduction

The main idea of this project will revolve around the use of Deformable Convolutions for an image classification problem. More specifically, we will be looking at the first version of Deformable Convolutions introduced in 2017.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning models specially designed for visual data processing. They consist of convolutional layers that detect patterns like edges and textures, pooling layers that downsample spatial dimensions, and fully connected layers that make high-level predictions. CNNs excel in tasks such as image recognition and classification, owing to their ability to automatically learn and extract hierarchical features from input data. These networks have significantly advanced computer vision applications, playing a crucial role in diverse fields such as image analysis, autonomous systems, and medical diagnostics. CNNs can capture patterns and relationships not easily seen by humans within visual data, making them a very popular method to use to understand and interpret images.

CNNs are inherently limited to model large, unknown transformations. The limitation originates from the fixed geometric structures of CNN modules: a convolution unit samples the input feature map at fixed locations; a pooling layer reduces the spatial resolution at a fixed ratio; a RoI (region-of-interest) pooling layer separates a RoI into fixed spatial bins, etc. There lacks internal mechanisms to handle the geometric transformations. This causes noticeable problems. For one example, the receptive field sizes of all activation units in the same CNN layer are the same. This is undesirable for high level CNN layers that encode the semantics over spatial locations. Because different locations may correspond to objects with different scales or deformation, adaptive determination of scales or receptive field sizes is desirable for visual recognition with fine localization[1]. We aim to address these limitations with the use of Deformable Convolution layers.

Related Work

Deformable Convolution

Deformable Convolution was developed as a solution to address the limitations of traditional CNNs in handling large unknown transformations, as mentioned in the Deformable Convolution V1 paper. By introducing offsets in the convolution operation, Deformable Convolution allows for free-form deformation of the sampling grid.

The offsets are obtained through a normal Convolution layer that takes the previous feature maps as input. These offsets are then utilised in the Deformable Convolution layer to determine the new sampling locations for performing convolution.

This approach enables convolution to be applied on local areas of interest, rather than using regular strides across the entire input.

In the original paper[1], the authors conducted experiments on the VOC 2007 test dataset, which is used for object detection. They replaced the last 1, 2, 3, and 6 3x3 convolution layers in the ResNet-101 architecture.

However, in their research, they did not explore the best location or configuration in replacing normal convolution with deformable convolution. Hence we will be diving deeper into the proper and efficient usage of deformable convolution.

ResNet 50

The ResNet architecture was used in the original Deformable Convolution paper. The ResNet models are based on the residual learning framework, which enables the training of extremely deep networks by introducing skip connections. Skip connections were introduced to address the problem of vanishing gradients that can occur in very deep neural networks such as ResNet. These connections allow the network to skip one or more layers and directly propagate information from earlier layers to deeper layers. Specifically, instead of directly feeding the output of one layer to the subsequent layer, the output is added element-wise to the output of a layer located further in the network. This helps facilitate gradient flow during training, alleviate the vanishing gradient problem, and enable the integration of low-level and high-level features.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 1: Characteristics of different ResNet models

Methodology

Datasets

We used 2 datasets as part of our project, the FashionMNIST image classification dataset[2] and the CIFAR10 image classification dataset[3]. The FashionMnist dataset has 60k rows for training and 10k rows for testing while the CIFAR10 dataset has 50k rows for training and 10k rows for testing. Both datasets were built-in in pytorch and are accessible using torch.datasets. Training is done on 80% of the train dataset and the other 20% is used as the validation dataset.

Pre-Processing

The ResNet50 model in PyTorch, like many other pre-trained models, was trained on ImageNet with images of varying sizes. As stated by Pytorch[4], our input images should be transformed in the same way as the images the model was trained on. This includes resizing the images to the appropriate size of 224 x 224, and normalising the pixel values to the range [0, 1]. The images from both datasets should be normalised using the mean and standard deviation values of [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225] respectively for the RGB channels. However, the FashionMNIST dataset(Grayscale Images) requires extra handling to increase the number of channels for each image from 1 to 3 before it is normalised.

Experiment 1

Deformable Convolution vs Normal Convolution

The first question to answer is whether the Deformable Convolution Layer does indeed outperform the normal Convolution Layer under similar training conditions on both the FashionMNIST and CIFAR10 data sets. We started by training a model with only normal convolutional layers and then we trained a model by replacing a few normal convolutional layers with deformable convolutional layers. We also adjusted each model by replacing different numbers of layers with Deformable Convolution layers

This experiment was carried out with reference to the paper mentioned above[1]. Tests conducted were slightly different but had the same goal.

Experiment 2

Exploring Best Configurations

The second experiment aims to see how replacing the 3x3 normal convolution layer at different blocks with a deformable convolutional layer affects the performance of the model. However, in this experiment, we would be individually testing each block and replacing more layers from each block to see if there are any obvious changes between each test. We conducted 4 experiments, each replacing normal convolutional layers with deformable convolutional layers from a different block.

Training

The training methodology for both experiments is as follows:

1. Training phase 1
    a. LR 0.001
    b. Max epochs 50
    c. Early stopping with patience 5
    d. Unfreeze offset , deformable convolution layer and fully connected layer
2. Training Phase 2
    a. LR 0.0001
    b. Max epochs 50
    c. Early stopping with patience 5
    d. Unfreeze offset, deformable convolution layer and fully connected layer

These training methods are applied to every variant of the ResNet50 model that we have trained to ensure a fair comparison between the test accuracies of all models.

Results

Experiments are performed on the ResNet 50 model with different numbers of deformable convolutions.

Under the model column, there are 4 numbers in each row x x x x and they correspond to the blocks conv2_x,conv3_x, conv4_x and conv5_x in Figure 1 respectively. The value in each position represents the number of 3x3 normal convolutional layers to be replaced by the deformable convolutional layer for that corresponding block. For example, Model 0 0 0 1 is a variant of the original ResNet50 model where we replace the last 3x3 normal convolutional layer with a deformable convolutional layer from block conv5_x.

Experiment 1

From our results we observe that replacing normal convolutional layers with deformable convolution layers does not necessarily result in significantly better accuracy and it comes with the cost of higher computation cost as seen from Figure 2. The computation cost for each model is represented by the boxplot while the test accuracy is represented by the line plot. In fact, the test accuracy even decreased as seen from Figure 3 for CIFAR dataset when we replaced normal convolutional layers with deformable convolution layers. Both Figure 2 and 3 are plotted in this manner to help us visualise the trade-off between test accuracy and computational cost of training a more complex model. This suggests that more training might be required to fine tune the offset layer further to achieve a more significant improvement in test accuracy. However, due to time constraints we decided to standardise the training across all models to ensure a fair comparison.

Model	Dataset	Test Accuracy	Test Cross Entropy Loss	Avg Training Time per Epoch (secs)	No. of Trainable Parameters
0 0 0 0	Fashionmnist CIFAR10	85.39% 81.96%	1.6131 1.6505	146.59 118.43	23,528,522 (+0.00%)
0 0 0 1	Fashionmnist CIFAR10	85.85% 81.84%	1.6085 1.6560	156.78 128.30	23,611,484 (+0.35%)
0 0 1 1	Fashionmnist CIFAR10	85.89% 78.78%	1.6123 1.6883	191.64 153.94	23,652,974 (+0.52%)
0 0 0 2	Fashionmnist CIFAR10	85.05% 80.20%	1.6156 1.6770	171.56 140.34	23,694,446 (+0.70%)

Table 1: Results from experiment 1

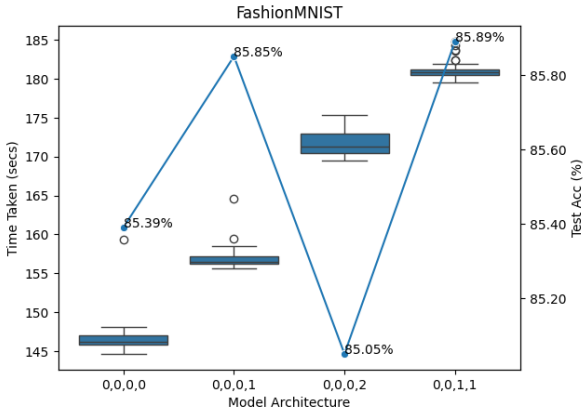

Figure 2: FashionMNIST results

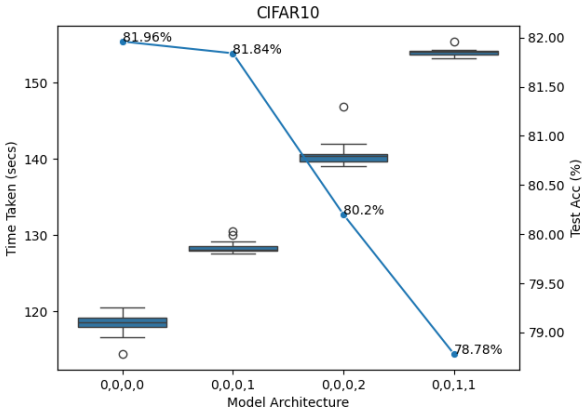

Figure 3: CIFAR10 results

Experiment 2

Based on the results for experiment 2 as seen in Table 2 and Figure 4 below, we observe that the test accuracy generally increases while the time taken per epoch decreases as we replace the normal convolutional layers in the deeper blocks. This observation is true except for the last block (conv5_x), where surprisingly the accuracy drops significantly. The line plot in Figure 4 illustrates this while the box plot shows the computational time. This result suggests it's recommended to replace the convolutional layers in the second last block(conv4_x) as it results in the best accuracy with a comparatively low computational cost as well.

Model	Dataset	Test Accuracy	Test Cross Entropy Loss	Avg Training Time per Epoch (secs)	No. of Trainable Parameters
3 0 0 0	Fashionmnist	80.06%	1.6644	399.52	23,559,680
0 3 0 0	Fashionmnist	82.41%	1.6464	275.20	23,590,784
0 0 3 0	Fashionmnist	83.00%	1.6417	211.13	23,652,992
0 0 0 3	Fashionmnist	75.16%	1.7288	183.02	23,777,408

Table 2: Results from experiment 2

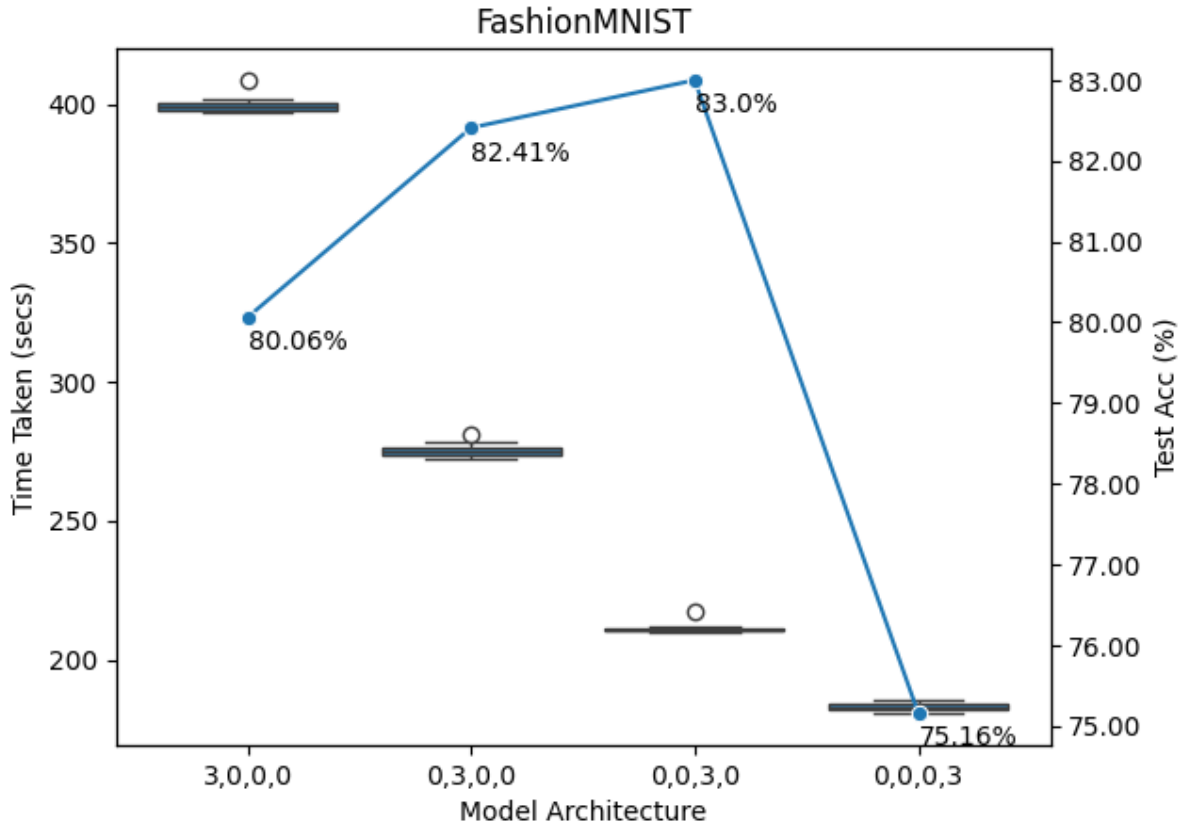

Figure 4: Results for different model architectures

Discussion

As shown in Figure 2, the ResNet model with the Deformable Convolution [0,0,0,1] does indeed slightly outperform the vanilla ResNet on the FashionMNIST dataset.

However for models with more than 1 layer of Deformable Convolution, we realised that with our simple fine tuning, it may not be sufficient to beat the vanilla ResNet. This could be attributed to the fact that we are introducing more parameters to train and a more complex model generally requires more training to achieve a good test accuracy. More advanced techniques such as learning rate decay or introduction of regularisation methods might be required to push the test accuracy higher. While the use of Deformable Convolution did yield slightly better performance, we need to consider the trade-off between the improvement in performance and the longer training time associated with it. We should also consider the possibility that the simplicity of the datasets, with objects being regularly shaped and positioned at the centre of the image, limits the full utilisation of the advantages offered by the offset layer.

In our second experiment, we replaced the last 3 layers of convolution in the different ResNet blocks with Deformable Convolution. Our results were quite surprising as it showed that replacing the last few layers did not actually result in the best performance. Instead, replacing the normal convolutional layers in the conv4_x block results in the best performance. We hypothesise that in the last block, the information has been compacted and aggregated through the many layers of normal convolution, hence there may not be a need to find the ideal location to sample through the offsets. Whereas for the other blocks(conv2_x, conv3_x, conv4_x), information is still sparsely spread out throughout the tensor, hence introducing the offsets in the middle allows the model to better focus on what information to aggregate.

For the front block (conv2_x), as information is still very sparse, it may be difficult for the offset to find the correct area to focus on. Hence normal convolution may still be better to first aggregate the information.

When we move to the middle blocks (conv3_x, conv4_x), the information is slightly more aggregated, thus offsets may work better in shifting the convolution.

Conclusion

In conclusion, the results from Experiment 1 shows that replacing normal convolutional layers with deformable convolutional layers in ResNet50 could potentially result in better test accuracy given sufficient training. The results from Experiment 2 was an interesting one which shows that replacing the last few convolutional layers of the ResNet50 model with deformable convolutional layers is not the most optimal choice in our case. This could potentially change the way in which future experiments are conducted to find the best configurations taking into account both accuracy and efficiency when applying deformable convolutions on ResNet models. Lastly, we acknowledge that our model could be better trained through employing different techniques such as training each deformable convolution layer by layer and introducing learning rate decay or other regularisation methods.

References

1. https://arxiv.org/pdf/1703.06211.pdf
2. https://www.kaggle.com/datasets/zalando-research/fashionmnist
3. https://www.cs.toronto.edu/~kriz/cifar.html
4. https://pytorch.org/hub/pytorch_vision_resnet/