Convolutional Neural Networks

Sudarshan Kulkarni

June 10, 2025

1 Dataset

We have about 30k images, with 256 categories. So an average of 120 images per category. Compared to larger datasets like in ILSVRC, which had an average of 1280 images per category, our data is quite smaller. It would definitely be beneficial to have more training data or fine-tune an existing model.

The clutter class containing a total of 827 images was deleted. Leaving us with 29780 images and 256 classes in total. Data were split into 0.8-0.2 for training and testing.

1.1 Data Augmentation

1. *torchvision.datasets.ImageFolder* is used to load the data from the folders, which automatically normalizes the pixel values. *torch.utils.data.random_split* to split into training and test set.

2. For training: Horizontal flips were added, but no vertical flips as there are a few classes for which it does not make sense.

3. Also, a random affine transformation is added, with image resizing to 224x224 after that.

4. There was also normalization with mean = std = 0.5 performed, but not for all models. And there was also resizing to bigger img (256 or 324) then taking a random crop (224 or 299) (this also was only for few models)

5. For test data, the image was resized to 256x256, and 10-crop testing was done with each of size 224x224. (In hindsight, the 10-crop test was very important here, because without it the model was actually performing very badly, as discovered by Grad-CAM)

2 What did I do?

Originally wanted to try out mixing Inception and Resnets, with DepthwiseSeparable convolutions, which would've allowed deeper networks with lower parameters. Found out about Xception model, which was quite similar, so most experimentation was done with it. I still tried out Resnets just to get another architecture to compare with. And having different architecture is beneficial when doing ensembling.

2.1 Fine tuning Xception (pre-trained)

I didn't spend too much time on this, as it was out of scope of submission. Loaded the default architecture, changed the last layer to output 256 logits instead of 1000, and retrained JUST that layer. Ran the model for 2 epochs, giving 85.6% top-1 accuracy and 96% top-5 accuracy. No self-trained model gave that much accuracy.

2.2 Custom Xception Model (1.9M)

Similar architecture of base model, but used 5 Middle Flows instead of 8. Each middle flow layer had 256 channels instead of 728, due to contraint on parameters. The final output was just global average pooled and fed into 256-softmax layer.
Probably should've trained for longer period but didn't think at that time. Got 51% accuracy on train set, 45% top-1 and 67.8% on top-5.

2.3 Custom Xception Model (5.9M)

Almost same as above, but the middle layers had 512 channels instead, and had just 1st block from Exit Flow, then global avg pooled and connected to 256-softmax layer.

This was trained for much longer, throughout the night. Only after this did I realise that, its not that dataset or my models are bad, but that I'm being too impatient. Sadly this was towards the end of the week, so didn't get to try out training for long time for many other models that I would've liked.

This got training set error of 87.2%, and val set 52% accuracy. I had not applied regularization before this. So I applied dropout and weight decay, but didn't get much time. Ran for only 4 epochs. Val set had 55.6% top-1 and 76.5% top-5 accuracy.

2.4 Resnet26

Similar structure to Resnet18, just every basic block replaced with a bottleneck block. Trained this for quite sometime, to reach 58% top-1 accuracy, 78% top-5 accuracy.

2.5 Resnet18

Didn't write its architecture from scratch, but trained it without pre-loaded weights. Did this at start to get an idea of how big of a network might be needed. But only trained for 11 epochs, so didn't get proper idea.

Got top-1 error of 40% and top-5 of 63.7%

2.6 Ensembling of models

This was the more interesting aspect ig. Took Xception (5.9M) and Resnet28, and added their raw logits together at the end, and used that for prediction (remember that both processed 10 images from 10-crop test).

This gave a result of 64% top-1 error, 82% top-5 error. If more time was put, and model(s) were trained for some longer time, much better results can be achieved.

3 Grad-Cam

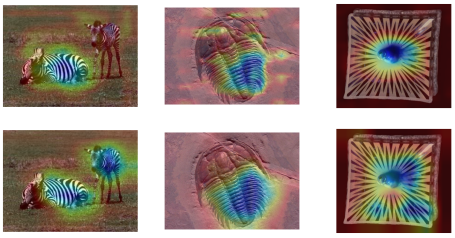

Figure 1: Row 1: Images that were correctly classfied by Xception(5.9M).
Row 2: Images that were correctly classified by Resnet26.

(figure 1) We can see how both the model is highlighting the objects in the images, showing what they focus on while predicting. What about the images that were incorrectly classified, that's more important. We compute $L^c_{grad-CAM}$ for both classes, the true and the predicted one.

3.1 Xception(5.9M)

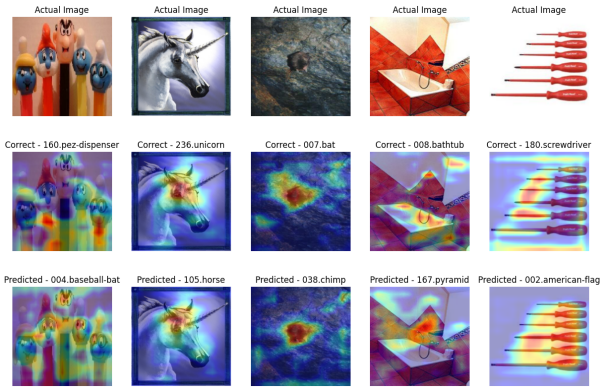

Figure 2: Images classified by Xception(5.9M)

(figure 2) It can be seen that the mistakes made can be fixed by more diverse dataset or pre-training on larger datasets. Because the the model needs to understand more finer structures in an image, its not distinguishing horns, the pattern on the walls or individual screw-drivers from the full group.

The prediction done by the model are also classes which are similar to correct class, or the images look similar.

3.2 Resnet26

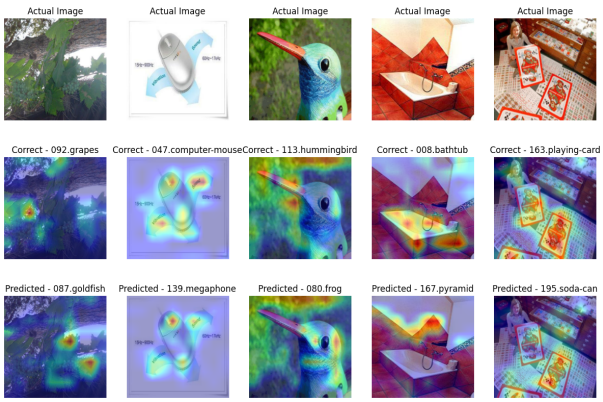

Figure 3: Incorrectly classified images by Resnet26

(figure 3) We can see for bathtub, for example, how the Grad-CAM activation with respect to bathtub class highlights the bathtub, and for pyramid, it highlights the design on the wall. This example, in my opinion, best highlights the use of Grad-CAM.