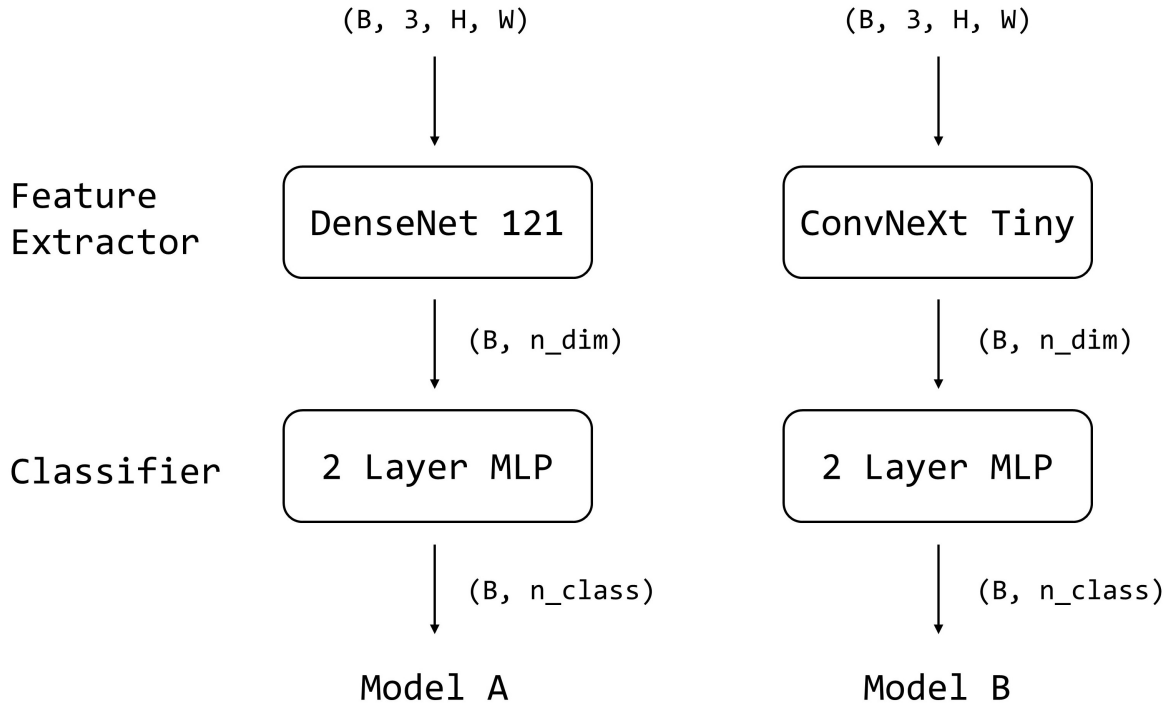
DLCV - Homework 1

Name: Chou Tun Hsiang **Student ID:** R11922163 **Date:** Sep 26 2022

1 Image Classification

1.1 Network Architectures



1.2 Accuracy

- Model A: 0.6416
- Model B: 0.9052

1.3 Implementation of Model A

- Optimizer: stochastic gradient descent
- Momentum: 0.9
- Weight decay: 2e-5

- Loss function: cross entropy
- Epochs: 50
- Learning rate: 0.1
- Learning rate scheduler: cosine annealing, proposed in SGDR: Stochastic Gradient Descent with Warm Restarts. The learning rate is computed as

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{T_{cur}}{T_{max}}\pi\right)\right)$$

- Batch size: 128
- Auto augmentation proposed in AutoAugment: Learning Augmentation Strategies from Data, followed by random erasing
- Label smoothing: 0.1

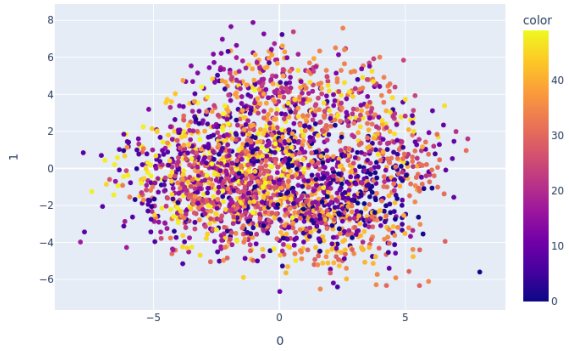
1.4 Comparison between Model A and Model B

The training hyperparameters of model B (ConvNeXt) is similar to that of model A (DenseNet), except that ConvNeXt is initialized with weights pretrained on ImageNet and trained for 20 epochs only.

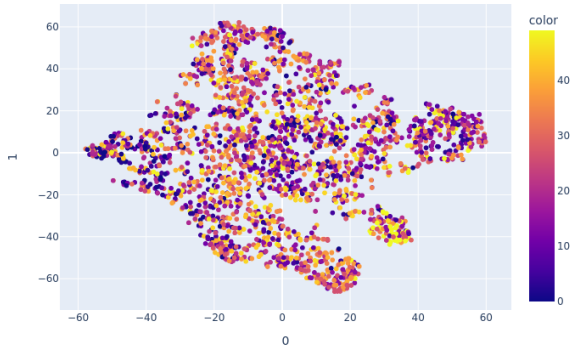
In terms of network architecture, both DenseNet and ConvNeXt are inspired by ResNet. To alleviate the vanishing-gradient problem, DenseNet connects each layer to every other layer in a feed-forward fashion. On the other hand, ConvNeXt follows different levels of designs from a Swin Transformer while maintaining the simplicity as a standard convolution network. The improvements include changing stage compute ratio, grouped convolution as proposed in ResNeXt, inverted bottleneck, larger kernel size, replacing ReLU with GELU, and subtituting batch normalization with layer normalization.

1.5 Visualization

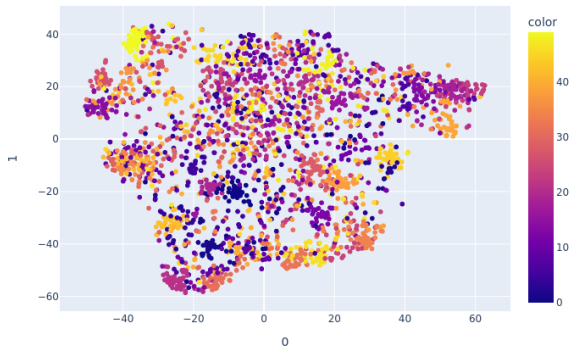
Figure 1 visualizes the extracted features. From the figures we can observe that PCA might not be a proper choice for feature visualization in this case. Data points from distinct classes mingle on the 2d plane. On the contrary, t-SNE shows that the model is actually learning how to differentiate the images. At the first epoch, all the points are still mixed up together. As the training goes on, some clusters appear. After training for 50 epochs, clearly we can tell from the distribution that some classes are more alike than others, such as the orange and purple points around position (40, -20) may share similar features.



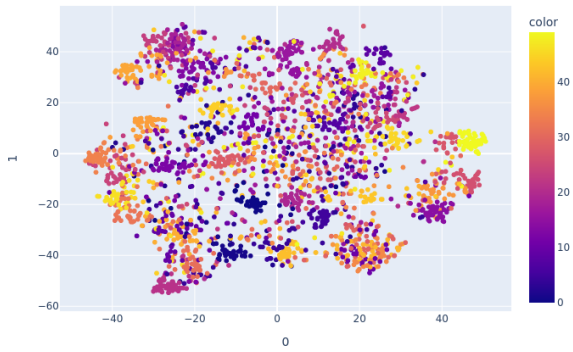
(a) PCA, 50 epochs



(b) t-SNE, 1 epoch



(c) t-SNE, 25 epochs



(d) t-SNE, 50 epochs

Figure 1: Visualization of extracted features

2 Image Segmentation

2.1 VGG16-FCN32s Architecture

Figure 2 shows the architecture of VGG16-FCN32s, where VGG16 acts as the backbone. The last three fully connected layers in the original VGG16 are replaced with two convolution layers and one transposed convolution layer to upsample the features back to the size of the input image, so that we can make pixel-wise predictions.

2.2 ResNet101-FCN Architecture

The improved model (figure 3) is also a FCN network, but with ResNet-101 as backbone. Model pretrained weights are available on Pytorch, which was trained on a subset of COCO, using only the 20 categories that are present in the Pascal VOC dataset. Within the FCN head, batch norm, ReLU, and dropout layers are inserted between two convolution layers.

Compared to model A (VGG16-FCN32s), bilinear interpolation is used instead of transposed convolution for upsampling. Since the number of classes is different from the pretrained scenario, an additional point-wise convolution is taken to convert the ouput from 21 channels to 7 channels.

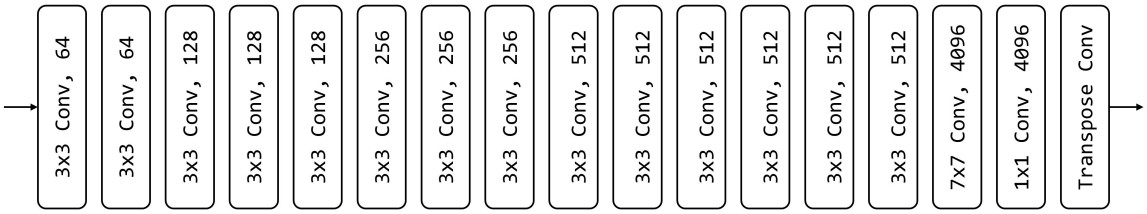


Figure 2: VGG16-FCN32s architecture

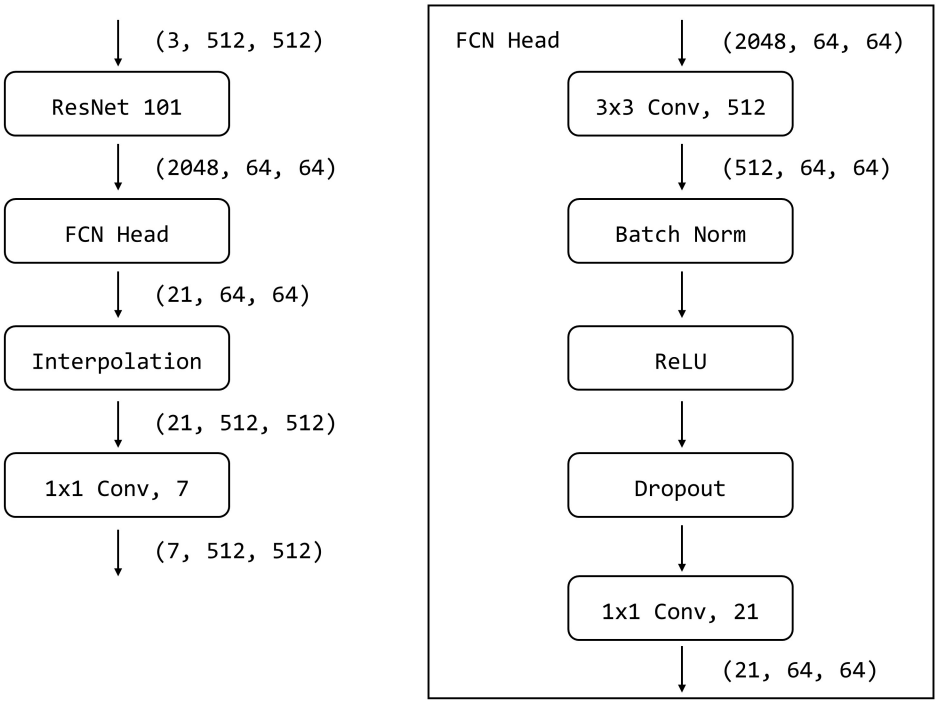
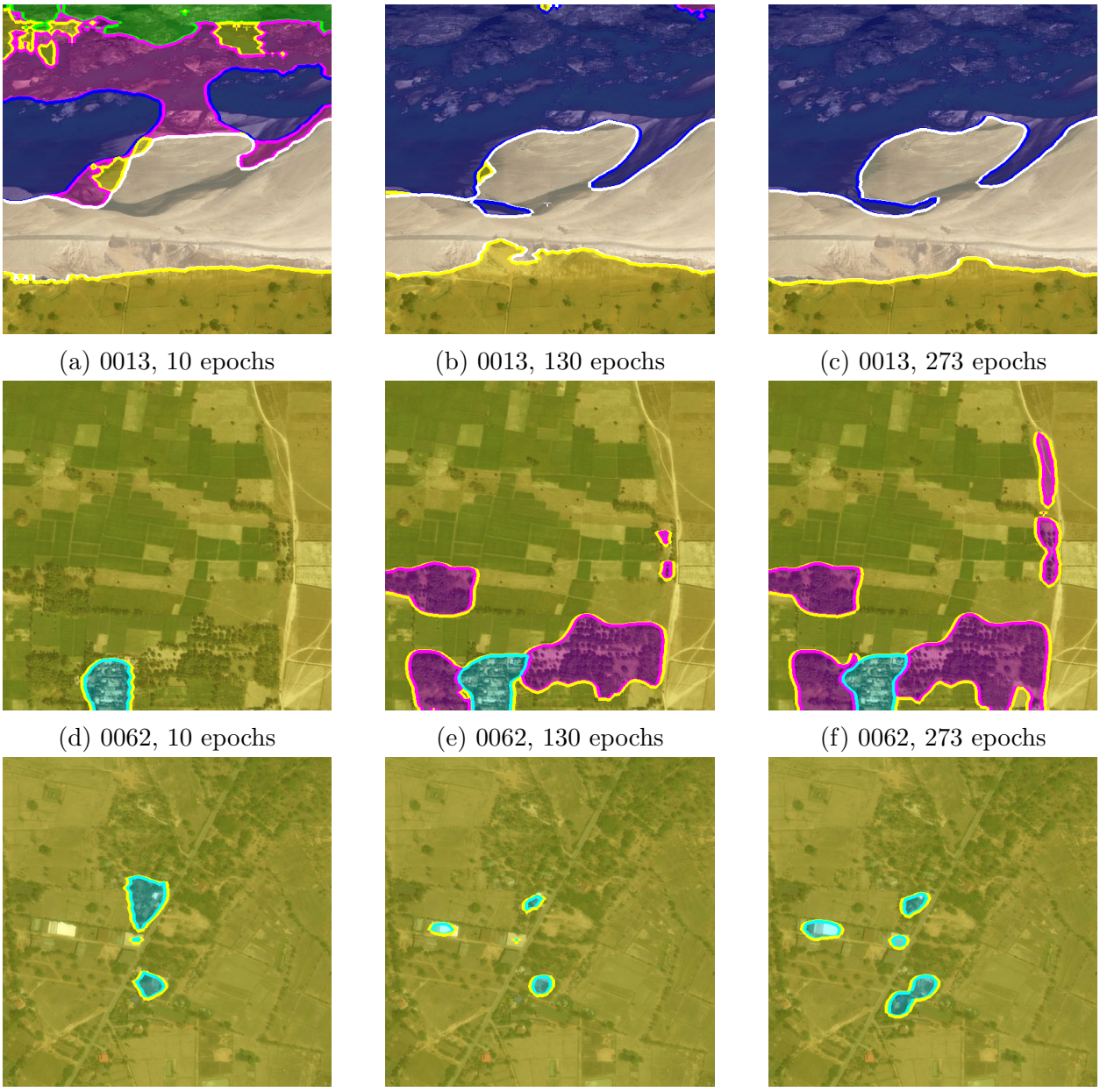


Figure 3: ResNet101-FCN architecture

2.3 mIOUs

- VGG16-FCN32s: 0.6927
- ResNet101-FCN: 0.7475

2.4 Visualization



(a) 0013, 10 epochs (b) 0013, 130 epochs (c) 0013, 273 epochs

(d) 0062, 10 epochs (e) 0062, 130 epochs (f) 0062, 273 epochs

(g) 0104, 10 epochs (h) 0104, 130 epochs (i) 0104, 273 epochs

Figure 4: Visualization of segmentation masks