Image Competion on CIFAR-10

Mason Swofford
mswoff@stanford.edu

Backround

Image completion, or image in-painting, is a new and rapidly growing area of research in computer vision. It involves taking an image with some pixels removed or obscured, and then trying to guess the missing pixel values, so that the in-painted image matches the original. Current state of the art models employ GANs to learn a distribution for the missing pixels; however, these models are computational expensive and hard to train. As well, they can typically only be used on specific image classes (e.g. human faces, natural scenery, etc.) For this project, more traditional convolutional networks were employed.

Data

CIFAR-10: 60,000 32x32 pixel RGB images from 10 different classes

- 8x8 mask applied to the center of every image to create an input X and ground truth Y
- The ground truth Y is an 8x8x3 image which is flattened into a 192 dimensional vector

Original Inage

Input X to Network

Ground Truth Y

Models

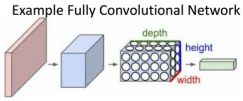
Various convolutional models were employed to test how different architectures affect performance. Mean squared error was used as the cost function for all models

Fully Convolutional Network

The first models tested were various fully convolutional networks, made up of only convolutional and max pooling layers. Fully convolutional networks have been shown to be effective for image segmentation problems, as they maintain spacial information in the image. Three different sized networks were tested and the input sizes at each layer are listed below. Note: all layers are valid convolutions with rectified linear activation unless otherwise noted.

- Shallow Network: 32x32x3 -> 28x28x10 -> 24x24x20 -> 12x12x20 (Max Pooling) -> 8x8x20 -> 8x8x3 -> 192x1 (Flatten)
- Deep Network: 32x32x3 -> 28x28x20 -> 24x24x40 -> 12x12x40 (Max Pooling) -> 10x10x60 -> 8x8x80 -> 8x8x3 -> 192x1 (Flatten)
- Super Deep Network: 32x32x3 -> 32x32x40 (7x7 Kernel "Same" Convolution) -> 32x32x40 (7x7 Kernel "Same" Convolution) -> 26x26x40 -> 20x20x60 -> 16x16x60 -> 8x8x60 (Max Pooling) -> 10x10x60 -> 6x6x60 -> 2x2x48 -> 192x1 (Flatten)

Example Fully Convolutional Network



Convolutional Network with Fully Connected Layers

The next model used was a convolutional network with two fully connected layers at the end. These types of networks are often used in image classification problems.

- Fully Connected Network: 32x32x3 -> 28x28x10 -> 24x24x20 -> 12x12x20 (Max Pooling) -> 8x8x30 -> 6x6x40 -> 1,440x1 (Flatten) -> 768x1 (Fully Connected Layer) -> 192x1 -> (Fully Connected Layer)

Encoder Decoder Network

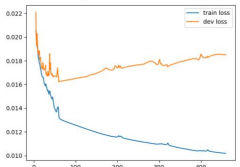
The last models tried were two encoder decoder networks. These networks first perform convolutions to decrease the size of the image and create a small dimensional "encoding" of the image. They then up-sample the encoding using deconvolutions to create the output image.

- Encoder Network: 32x32x3 -> 28x28x10 -> 24x24x20 -> 12x12x20 (Max Pooling) -> 8x8x30 -> 4x4x128 -> 6x6x3 (3x3 Kernel Deconvolution) -> 8x8x3 (3x3 Kernel Deconvolution) -> 192x1 (Flatten)
- Deep Encoder Network: 32x32x3 -> 28x28x30 -> 24x24x40 -> 12x12x40 (Max Pooling) -> 8x8x40 -> 4x4x40 -> 1x640 (Flatten) -> 1x768 (Fully Connected Layer) -> 2x2x40 (3x3 Kernel Deconvolution) -> 4x4x40 (3x3 Kernel Deconvolution) -> 8x8x3 (3x3 Kernel Deconvolution) -> 192x1 (Flatten)

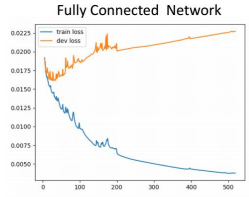
Results & Error Analysis

Here are the train and dev losses vs epoch for select networks.

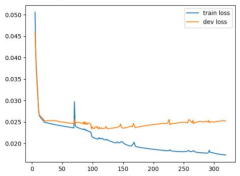
Deep Encoder Network



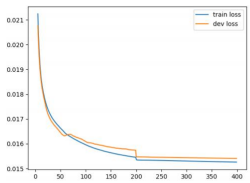
Fully Connected Network



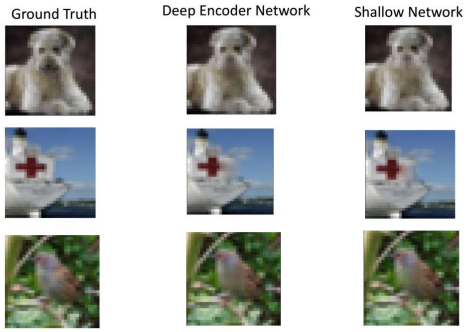
Deep Convolutional Network



Shallow Network



Example Outputs



Conclusions

- As expected, the deeper models were able to over fit the training set; so early stopping had to be used to get the best performing models
- In particular, the models with fully connected layers (encoder networks and fully connected networks) quickly over fit the training set.
- Surprisingly, the shallow network was able to perform similar to the deeper models on the development set
- The models were able to perform reasonably well on the test set. They tended to blur the image; so they did not perform well on very precise images which required straight lines to look natural (e.g. the cross on the flag).

Future Steps

- With more compute and time, it would be interesting to try these same models on a larger dataset of larger images (e.g. ImageNet 64x64 RGB dataset). On a larger dataset, the deeper networks would likely outperform the shallow network, as they can learn more complex mappings and benefit from more data
- It would be interesting to also train a GAN on these small datasets to see how well the state of the art performs with limited data and small images
- It is also possible that these simple networks would perform better if they were only learning one class of images (e.g. faces)

References

CS231n Convolutional Neural Networks for Visual Recognition. http://cs231n.github.io/convolutional-networks/

Amos, B. (2018). Image Completion with Deep Learning in TensorFlow. http://bamos.github.io/2016/08/09/deep-completion/ [Accessed 21 Mar. 2018].

Long, Shelhamer, Darrel (2014). Fully Convolutional Networks for Semantic Segmentation. http://arxiv.org/abs/1411.4038