SuperTiny Resoluton: *Super Resolution Models in the Tinygrad Machine Learning Framework*

Garrett Partenza
Khoury College of Computer Sciences
Northeastern University
Boston, MA
partenza.g@northeastern.edu

Jiameng Sun
Khoury College of Computer Sciences
Northeastern University
Boston, MA
sun.jiam@northeastern.edu

***Abstract*— We aimed to build a super-resolution model to augment the quality of satellite imagery in the Tinygrad machine learning framework, a new machine learning framework whose core codebase remains under 1000 lines of code. We contribute to the opensource framework by implementing sub-pixel convolution inside using only the frameworks NumPy core as to not break the gradient tree. Our implemented architectures included many popular super resolution models including SRCNN, ESPCNN, and Transformer. Our model results failed to produce high-resolution images within the allocated training time.**

Keywords—super resolution, machine learning, computer vision, convolutional neural networks

I. Introduction

The WorldStrat dataset is the latest satellite image dataset of all kinds of landforms. It contains not only low-resolution images from sentinel 2 but also high-resolution images of the same area from Airbus. The image pairs make the dataset suitable for training a super-resolution model for satellite imagery. In project three, we implemented a vision transformer super-resolution model and trained it with the dataset. The outputs were very noisy, so we decided to improve the model as our final project. The vision transformer implemented in project three consisted of a convolutional layer and a transformer module. We hypothesized that the simple convolutional layer responsible for embedding patches couldn't extract meaningful features for the transformer module to learn effectively. To improve results, we attempted to augment the convolutional layer by implementing historically successful super resolution architectures, with the final goal of retraining our original convolution-transformer architecture to produce better results.

II. Related Work

Image super-resolution (SR) is the process of converting a low-resolution image with coarse details to a high-resolution image with better visual quality and refined details. This technique is useful for applications such as large computer displays, HD television sets, and hand-held devices like mobile phones and cameras. SR can be performed using a single image or multiple images. Our vision transformer and our focus for the final project is single image super-resolution (SISR). SISR can be divided into two main categories: traditional methods and deep learning methods. Recent algorithms tend to rely on deep learning models to reconstruct the missing details for accurate super-resolution. Our project focuses on the deep learning side.

There are five types of super-resolution deep networks classified based on the architectures. Linear networks use a simple structure of several convolution layers. Residual networks add skip connection in the network design. Recursive networks utilize recursively connected convolutional layers or recursively linked units and try to break down the harder SR problem into a set of simpler ones. Densely connected networks employ densely connected CNN layers. Attention-based networks use transformers to solve the super-resolution problem. Our vision transformer is an attention-based network. For the final project, we focus on examining different linear networks which will be used as the CNN module of the vision transformer.

For linear networks, there are two ways to handle upsampling: upsampling the input before it enters the deep network and upsampling with a CNN layer. For networks using the first way to handle upsampling, we studied SRCNN, VDSR, and DnCNN and chose to implement SRCNN. For networks using the second way, we researched about FSRCNN and ESPCN. We implemented the ESPCN.

SRCNN (Super-Resolution Convolutional Neural Network) is a convolutional neural network (CNN) designed for image super-resolution (SR). It is considered the first successful attempt at using only convolutional layers for SR, and it inspired many later attempts in this direction. The SRCNN architecture is made up of three convolutional layers and two Rectified Linear Unit (ReLU) non-linearity layers. The LR input patches are first downsampled and then upsampled using bicubic interpolation to match the size of the HR output image. The network is trained to minimize the difference between the output and the ground truth HR images using a mean squared error (MSE) loss function.

VDSR (Very Deep Super-Resolution) is a deep convolutional neural network (CNN) based on the architecture of the VGG-net, which uses fixed-size convolutions (3×3) in all network layers. To avoid slow convergence in deep networks, VDSR proposes two strategies. First, instead of directly generating an HR image, it learns a residual mapping that generates the difference between the HR and LR images. This provides an easier objective and allows the network to focus on only high-frequency information. Second, gradients are clipped within a range, which allows very high learning rates to speed up the training process.

DnCNN (Denoising Convolutional Neural Networks) is a convolutional neural network designed for image super-resolution and image restoration. Instead of predicting the latent super-resolved image, DnCNN learns to predict a high-frequency residual directly. The DnCNN architecture is similar to SRCNN, consisting only of convolutional, batch normalization, and ReLU layers. Although DnCNN and SRCNN were able to report favorable results, they are computationally expensive due to the batch normalization operations after every convolutional layer.

FSRCNN (Fast Super-Resolution Convolutional Neural Network) consists of four convolution layers and one deconvolution. Though the architecture of FSRCNN is similar to the SRCNN, the input for FSRCNN is the original low-resolution image. This coupled with smaller filters used by FSRCNN helps it become more efficient than SRCNN. FSRCNN upsamples the inputs with the last CNN layer.

The ESPCN (Efficient Sub-Pixel Convolutional Neural Network) is a fast approach to single image super-resolution (SR) that can operate in real-time for both images and videos. Traditional SR techniques first map the low-resolution image to a higher resolution using interpolation, and then learn the SR model in the higher dimensional space. This results in high computational requirements. ESPCN proposes instead to perform feature extraction in the LR space and use a sub-pixel convolution layer at the end to aggregate the LR feature maps and simultaneously perform projection to the high-dimensional space to reconstruct the high-resolution image. This significantly reduces the memory and computational requirements.

III. Dataset

Our project utilized two datasets. The first, WorldStrat, provides high-resolution and the corresponding low-resolution satellite imagery of nearly 10,000 km^2, and represents the target dataset of our end-goal architecture. The unique locations of the images ensured a stratified representation of all types of land use all over the world. The dataset provides the following four types of satellite images: 12-bit radiometry high-resolution images in their raw format, downloaded directly from Airbus, 8-bit radiometry high-resolution images, generated by converting both 12-bit variants down to 8-bits, 16 temporally matched low-resolution Sentinel-2 Level-2A revisits for each high-resolution image, and 16 temporally matched low-resolution Sentinel-2 Level-1C revisits for each high-resolution image. We used the RGB channels from both Airbus HR images and sentinel LR images to train our initial transformer-based architecture.

The second dataset is a simpler super resolution dataset obtained from the Kaggle competition website. The dataset consists of 100 low-high resolution image pairs, where low-resolution images are 96x96 and high-resolution images are 384x384. The images are of random objects including people and nature. We utilized this dataset as an initial testing dataset for our model architectures which required less complexity (allowing for less model parameters) and less compute (lower resolution images reduced training time). This dataset was obtained as a steppingstone to the more complex WorldStrat dataset.

IV. Methods

Our attempted implementations included five different methods: convolution-transformer, single-layer convolution, transposed-convolution, multi-layer convolution, and sub-pixel convolution, also known as pixel-shuffle. We also utilize three different interpolation methods for baselining results. Our convolution-transformer was trained on the WorldStrat dataset, while all other models were trained on the Kaggle dataset.

A. *Convolution-Transformer*

Our transformer-based architecture was our first implementation originating with project three. The model consisted of two sub-models: a patch-level convolutional embedder, and a transformer encoder-only module responsible for passing embeddings along different patches. The original architecture was trained to increase satellite imagery from 400x400 RGB to 1024x1024 RGB and breaks images into 20x20 non-overlapping patches. The model directly operated off low-resolution imagery as input without interpolation. convolutional layer embedded images into 128-dimensional feature vectors before being inputted into six transformer blocks each with eight attention heads. The output was reshaped and upscaled using transposed convolution to increase each patch to 64x64.

B. *Transposed-Convoliution*

Our transposed convolutional architecture takes low-resolution imagery as input and utilizes zero padding and single-stride convolution to increase image size from 96x96 to 384x384. The architecture utilized three consecutive transposed convolution layers which expanded channels from 3, to 64, to 32, and finally back to 3. Kernel sizes of each layer were 5, 3, and 5, respectively.

C. *Single-Kernel Convolutation*

Our single kernel convolutional architecture takes low-resolution imagery as input, uses bicubic interpolation to increase the image size from 96x96 to 384x384, and then uses single-stride convolution with zero padding to correct the incurred blur from interpolation. The layer did not expand channels and used a large kernel size of 13x13. The large kernel size was chosen because after interpolation, neighboring-pixel information from the original low-resolution input became further apart. A single convolutional layer on the 4x interpolated image of kernel size 13x13 is approximately 3x3 with respect to the low-resolution image.

D. *Multi-layer convoliution*

Our multi-layer convolution architecture implements SRCNN by taking low-resolution imagery as input, applying bicubic interpolation to increase the image size from 96x96 to 384x384, and applying three consecutive single-stride convolutional layers to correct the incurred blur from interpolation. The kernel sizes where 5, 5, and 3, respectively. Following the authors of SRCNN, features were expanded from 3 to 64, to 32, and back to 3.

E. *Sub-pixel convolution*

In response to the computational inefficiency of our multi-layer convolutional architecture, we increase computational efficiency by directly operating off low-resolution imagery without interpolation. We achieve this by implementing sub-pixel convolution or pixel-shuffle, originally from the ESPCNN paper. Our architecture consisted of three consecutive convolutional layers which first expanded channels from 3 to 64, to 64, to 48. The 48 output channels were then reshaped to upscale single pixels to 4x4 grids. The image was then reconstructed.

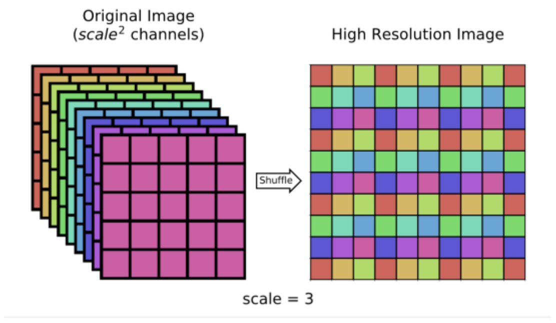

Fig. 1. *3x Sub-pixel convolution reshape visualized*

V. Experimental Setup

All models were trained on the Northeastern Discovery high-performance compute cluster. Models were trained with a batch size of 16 for 2000 optimization steps on an A100 GPU. ReLU activation was applied after all intermediate layers. Training times ranged from 30 minutes to 8 hours. Adam optimizer was used, and 2D Batch norm was applied after all intermediate layers. Gradient clipping was also applied with a limit of 1.0. The loss function was L2 (mean-squared error) between the output image and the ground truth high-resolution image.

VI. Results

Our architectures produced poor results. Convolutional-transformer and transposed convolution were unable to converge. We hypothesize that the transformer architecture 's patch embedding module failed to capture adequate features, and transposed convolution introduced too many artifacts due to the large-zero padding applied to upscale the image. Multi-layer convolution was unable to finish training due to computational inefficiency in its design. Single-layer and sub-pixel convolution architectures were able to converge, however, results were still not optimal.

The final test loss of the single-kernel model reached 0.15 (average pixel inaccuracy of 98) and the sub-pixel convolution architecture reached 0.67 (average pixel inaccuracy of 208) Example outputs for the single-layer and sub-pixel convolution architecture, as well as interpolation methods, are reported in Figures 2 and 3.

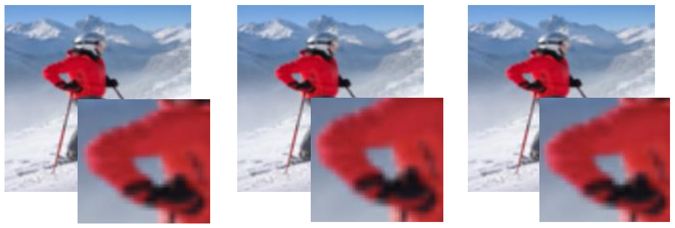

Fig. 2. *Nearest, Linear, and Bicubic interpolation methods on image 93 of Kaggle dataset.*

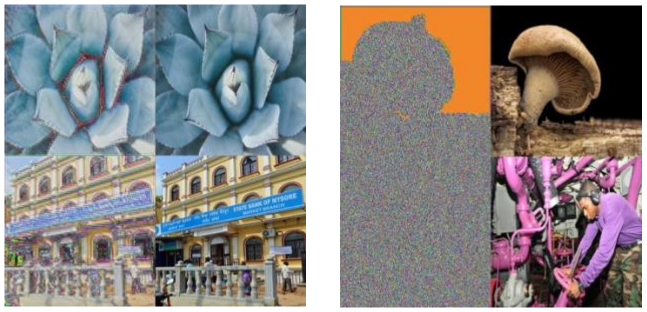

Fig. 3. *Train(top) and test (bottom) sample predictions at end of training for single-kernel (left) and sub-pixel (right) architectures*

Surprisingly, the single-kernel CNN actually provided decent outputs. We learned that the issue of our vision transformer didn't come solely from the CNN module. For future work, we need to further examine the transformer module in our vision transformer. The training process of the CNN models for super-resolution was much harder than we expected. With the same amount of training (2000 epochs for each), CNN models for other tasks such as digit recognition or color constancy would produce more satisfying results. We will request multiple GPUs from discovery HPC and train the CNN models longer to see if the outputs improve and obtain the good results presented in the original papers. After improving the CNN models, we will use the best one for our vision transformer. Our super-resolution vision transformer can be then used for scene classification and place of interest recognition tasks for the WorldStrat dataset.

Finally, if our vision transformer or CNN models produce ideal outputs and get approved by TinyGrad developers, they can be the first group of super-resolution models available for TinyGrad users.

References

[1] C. Dong, C. C. Loy, K. He and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 2, pp. 295-307, 1 Feb. 2016, doi: 10.1109/TPAMI.2015.2439281.

[2] W. Shi *et al*., "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1874-1883, doi: 10.1109/CVPR.2016.207.

[3] J. ICornebise, I. Oršolić, and F. Kalaitzis, "Open High-Resolution Satellite Imagery: The WorldStrat Dataset--With Application to Super-Resolution, " , 2022, *arXiv preprint arXiv:2207.06418*.

[4] S. Anwar, S. Khan, and N. Barnes, "A deep journey into super-resolution: A survey,", *ACM Computing Surveys (CSUR)*, *53*(3), 1-34.