Transformers For Recognition In Overhead Imagery: A Reality Check Supplemental Material

Francesco Luzi
Duke University
francesco.luzi@duke.edu

Aneesh Gupta
Duke University
aneeshgupta8@gmail.com

Leslie Collins
Duke University
leslie.collins@duke.edu

Kyle Bradbury
Duke University
kyle.bradbury@duke.edu

Jordan Malof
University of Montana
jordan.malof@umontana.edu

Abstract

Here we describe in greater detail the models we used and the modifications we made on these models for our study.

1. Benchmark Segmentation Models

For our experiments we evaluated three models, Unet [7], TransUnet [2], and SwinUnet [1], shown in Figure 1 (a), (b), and (c) respectively. We evaluated these models due to their similar architecture and performance in segmentation. We constrained all models to have a similar parameter count of approximately 105 million parameters.

Unet. The Unet architecture consists of a encoder (contracting path) and a decoder (expansive path). The encoder involves repeated applications of 2 3×3 convolutions with batch normalization [5] and without padding, followed by a ReLU activation. After several of these, the features are downsampled using a 2×2 max pooling layer with stride 2. Downsampling is done 4 times in the encoder, with skip connections to the decoder before downsampling. After the last down sampling stage, the features are processed again by several convolutional blocks, then given to the decoder.

The decoder up-samples the features using a 2×2 convolution ("up-convolution") that halves the number of features in the feature channel. These up-sampled features are then concatenated with the corresponding features from the encoder through the skip connection. The features from the encoder must be cropped since we use convolutions without padding, and thus the features from the encoder are of a larger spatial dimension. The feature block is then processed in the same manner as the encoder, with multiple feature blocks and 4 up sampling stages in total. In the final stage the model uses a 1×1 convolution to map all 64 channels to the segmentation mask. The final mask is smaller spatially than the input. To account for the reduction in size, the edges of the input are mirrored such that the resulting size is the same dimension as the original image.

For our work, we used a slightly modified version of the Unet that is compatible with the ResNet101 [4]. We made two changes to the network. We use a third layer in each convolutional block that includes padding. This retains all dimensions but adds parameters and representational power to the model. We also use a greater number of convolutional blocks in the encoder. This is done to increase the parameter count and take advantage of the ResNet101 pretrained weights.

SwinUnet. The SwinUnet Tiny follows a similar format to the Unet, with the exception of only having three downsampling layers. The encoder follows a $[2, 2, 2, 2]$ format, with 2 Swin transformer blocks before the first downsampling. This is repeated 3 times and then there are 2 final Swin transformer blocks before upsampling. For the decoder, SwinUnet follows the same $[2, 2, 2]$ structure. We modified this structure to better fit the Swin Base model and to increase the parameter count by changing the layer allocation. We kept the decoder the same, but changed the encoder to have a $[4, 4, 10, 3]$ structure. We had done some minor tests to determine what layer allocation resulted in the greatest performance on the validation set. We do note that our investigation was limited in architectures and in no way claim that this is the best way to restructure the SwinUnet. Pretrained weights were loaded in directly from the Swin Base checkpoint, trained on ImageNet [6]. In the case where a layer contained a greater number of transformer blocks than in the respective Swin Base model, such as in the first two layers, we randomly initialize the blocks with no corresponding pretrained weights. We found that even with this random initialization in some of the blocks, the $[4, 4, 10, 3]$ configuration performed best.

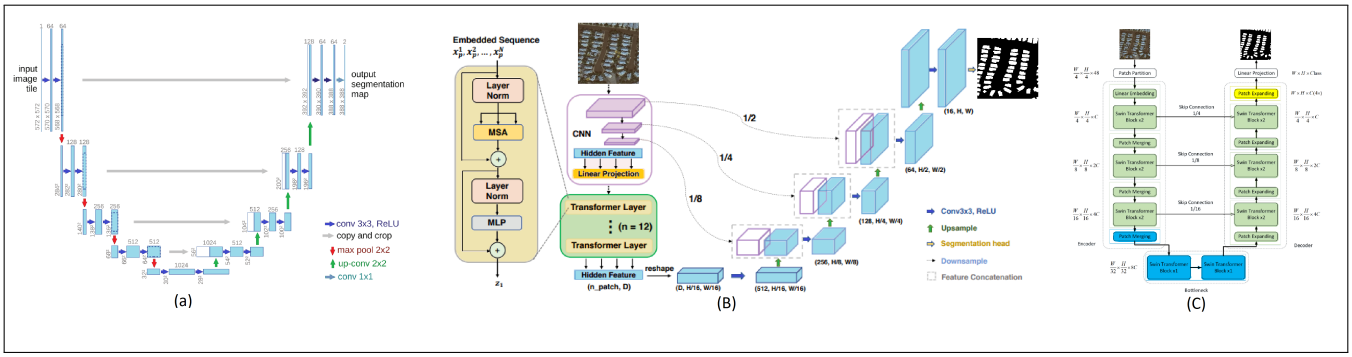

Figure 1. (a), (b), and (c) Show the architecture for the Unet, TransUnet, and SwinUnet respectively. These figures were inspired directly by [7, 2, 1], respectively

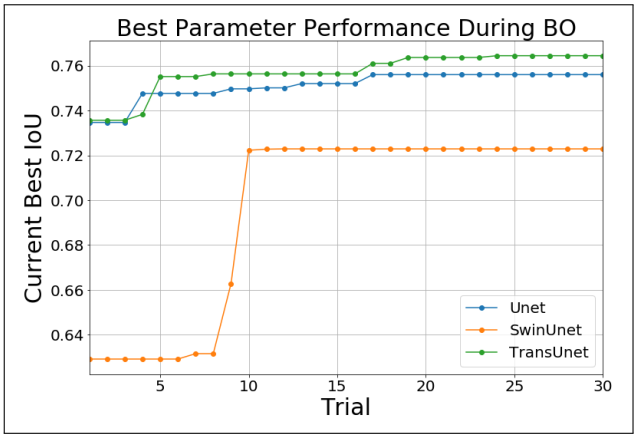

Figure 2. The max IoU trial versus the current iteration is displayed for all three baseline Bayesian optimizations. Each model improves greatly in the beginning but then converges at around or before the 20th iteration, with minor or no improvements in performance afterwards.

We also compare using the Swin Tiny version of the SwinUnet. This version follows the same structure as in the original paper and the official repository without modification.

TransUnet. We use the official code base and implementation of the TransUnet with no modifications for our primary results. The TransUnet uses a CNN-Transformer hybrid, with the transformer portion being Visual Transformer layers (ViT) [3]. The CNN uses convolutional blocks that follow the format of 1×1 convolution, 3×3 convolution, and then another 1×1 convolution. Each convolution is followed by a batch normalization layer and each block is ended by a ReLU activation. Padding is used so that the input dimensions are the same as the output. The CNN contains three down sampling layers, with skip connections to the decoder before downsampling. The output of the CNN portion is projected linearly into a two-dimensional matrix with 1×1 patch embedding. These are then processed by the transformer layers in a sequential manner.

The ViT transformer layers in the model perform global attention on the CNN features and use pretrained weights from ImageNet. We have several variants of the model with different numbers of layers. We keep all other aspects of the model the same and only remove layers from the end of the transformer sequence. In the 0 layer version of the TransUnet we pass the CNN features directly to the decoder after upsampling. For our fully connected version of the TransUnet we replace all transformer layers with fully connected layers of the same output and input size. We use layer normalization and ReLU activation between each fully connected layers. For the convolutional version of the TransUnet we replace the transformer layers with 3×3 convolutional layers and remove the patch embedding. We use batch normalization after each convolution and ReLU activation. We also add residual connections between convolutional layers as is commonly done in practice and found that it was beneficial to model performance.

The decoder in the TransUnet follows a similar structure to the encoder, with only 2 transformer blocks per spatial resolution. The features from the skip connections are concatenated and processed in the same manner as the Unet.

2. Bayesian Optimization

The conclusions of this work depends heavily upon the premise that BO found good hyperparameters for each of our competing models, reflecting their performance in practice, given a typical systematic optimization of model hyperparameters. In this section we provide some additional details that complement and elaborate on the information in the main body of this paper.

We present several pieces of information that collaborate our claim that the BO was effective. The first, which was also shown in the main body of work, is Figure 2 and Figure 3. These figures are presented here as a reference for the reader since they are unchanged from their prior form.

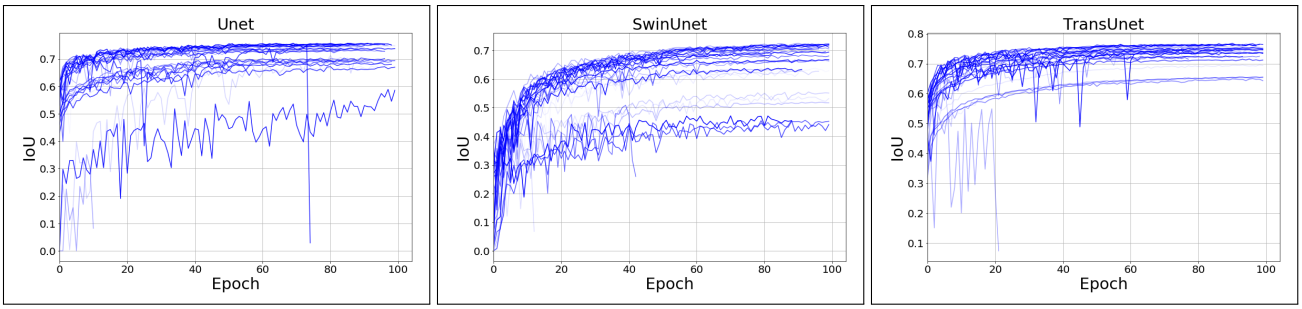

Figure 3. Shown here is the training validation at every epoch for each iteration of Bayesian optimization. Each model was allowed 30 iteration of Bayesian optimization to find the optimum hyperparameters. The darker lines represent the later trials and the lighter the line, the earlier the trial. Since we used an adaptive stopping criteria some trials end earlier if their performance stagnates or is very poor for that stage of training.

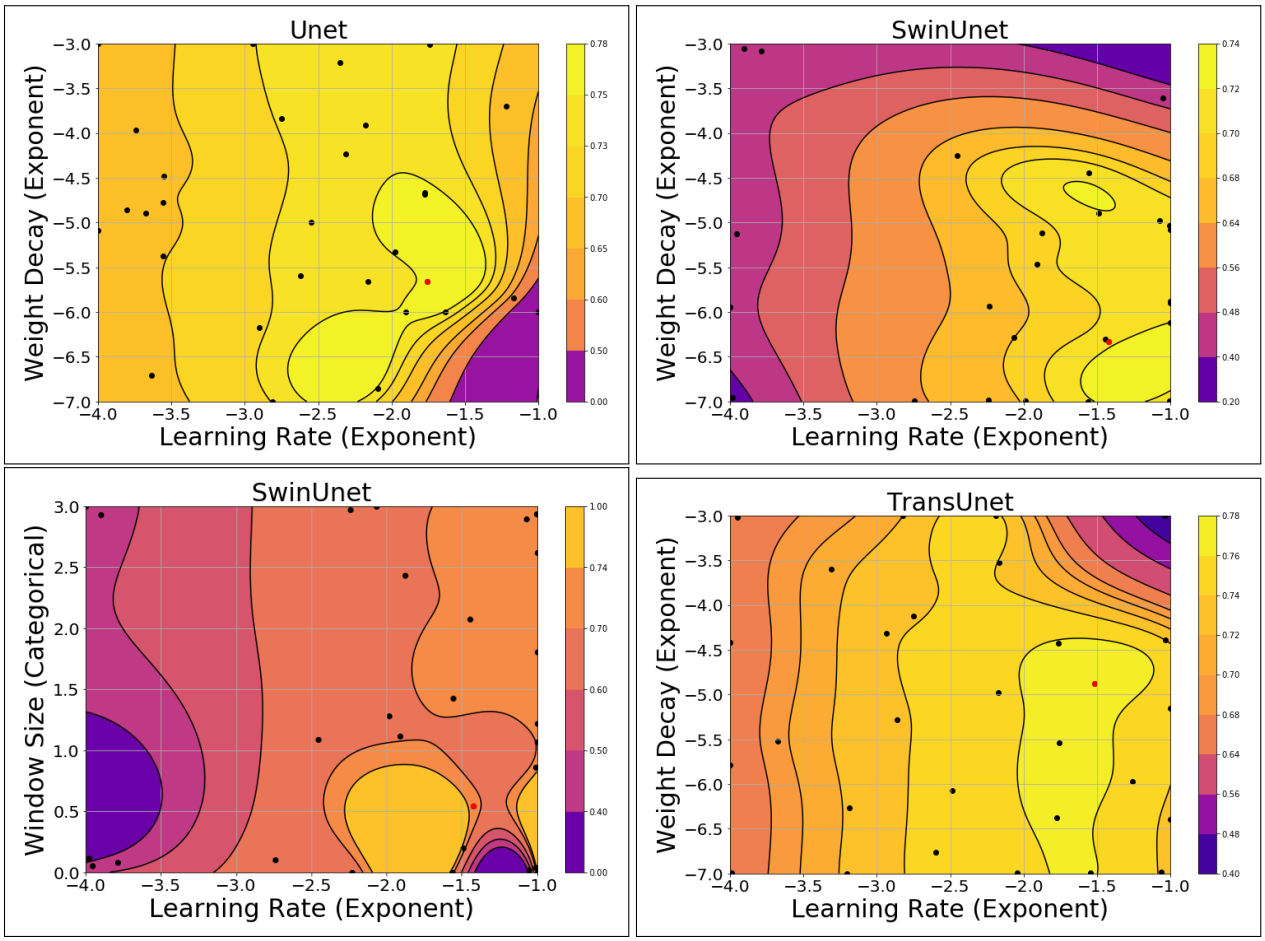

Figure 4. Heat maps of the parameter space searched with Bayesian Optimization. Sampled points are displayed as block dots, with the final parameter choice represented by a red dot, and Gaussian processes are used to model the points in between to fill out the space. Learning rate is plotted on the X-axis with weight decay plotted on the Y-axis with the exception of the first plot in the second row where window size is plotted on the Y-axis. Both learning rate and weight decay are sampled by their exponential (-3 gives a value of 10^{-3} or $1e-3$) so that there is equal weight given to parameter values on a $\log_{10}$ scale.

We also provide a more complete version of the heat map in Figure 4. Here we include the addition of the heat map for the window size in the SwinUnet optimization. This heat map was excluded from the main body of work due to

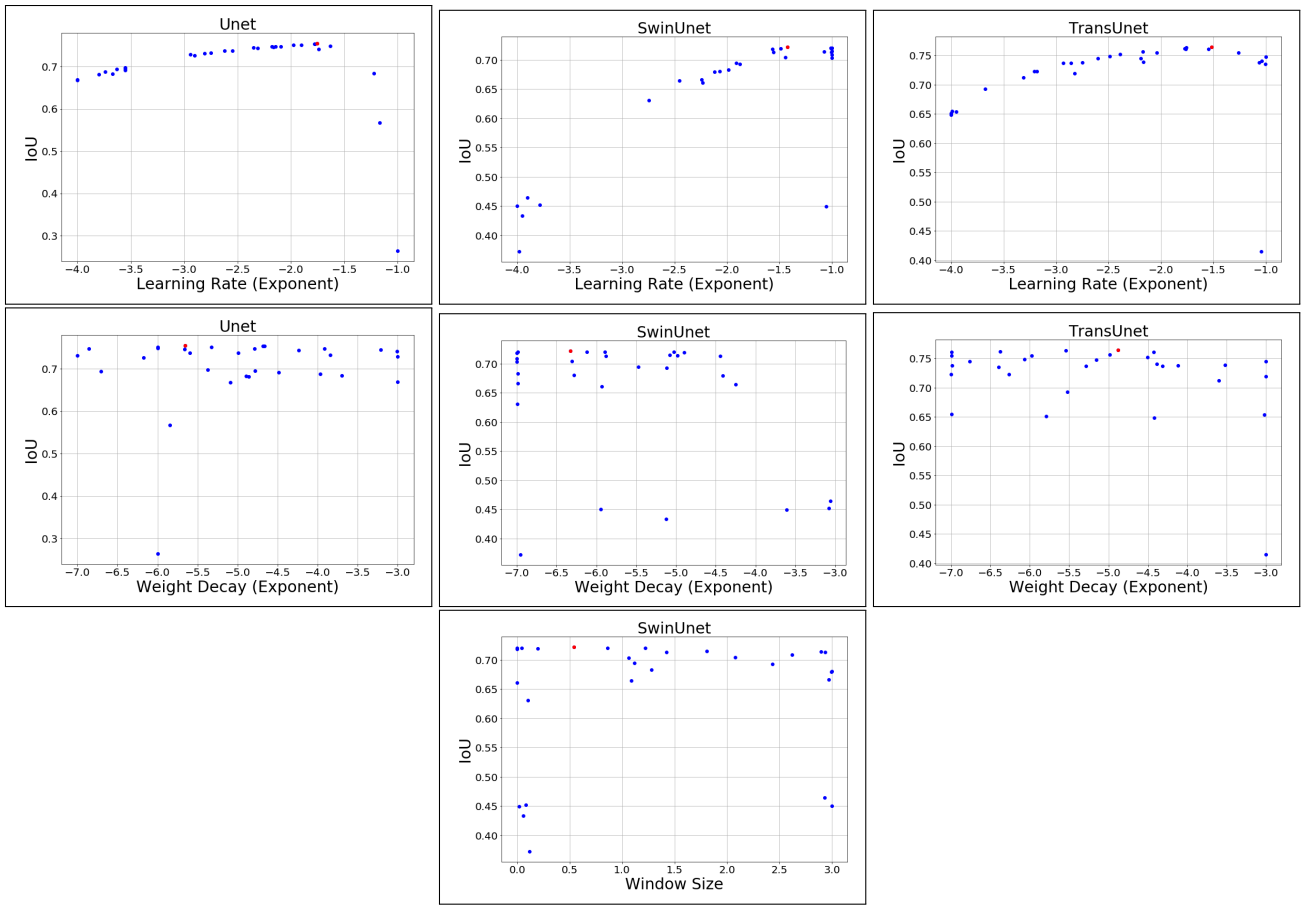

Figure 5. The searchable parameter versus the IoU for each Bayesian optimization and each parameter is displayed. The learning was found to be the most impactful parameter in the search and is thoroughly sampled with a clear relationship between the model parameter and performance. The learning rate and weight decay is sampled by it's exponential (-3 gives a value of 10^{-3} or $1e-3$) so that there is equal weight given to parameter values on a $\log_{10}$ scale. The window size is categorical with specific ranges being mapped to different window sizes.

space constraints and to simplify comparison between the models. The heat map for the window size in the SwinUnet is plotted against the learning rate since that was found to be the most influential parameter.

We also include in Figure 5 each parameter plotted against the resulting IoU from that trial. This gives a more complete understanding of the importance of each parameter and the density of the space that was explored for each parameter. From this figure we can clearly see that the learning rate was the most influential parameter on performance and that the other parameters had little effect that is difficult to determine on a one dimensional figure. This figure was also excluded from the main body of work due to space constraints.

3. Example Imagery

Due to space constraint we reduced the number of example segmentation outputs we presented in the main paper. We believe that example out is valuable for determining the qualitative performance of the model and can be helpful in understanding the difficulty of the problem and diversity of the data. For this reason we include Figure 6.

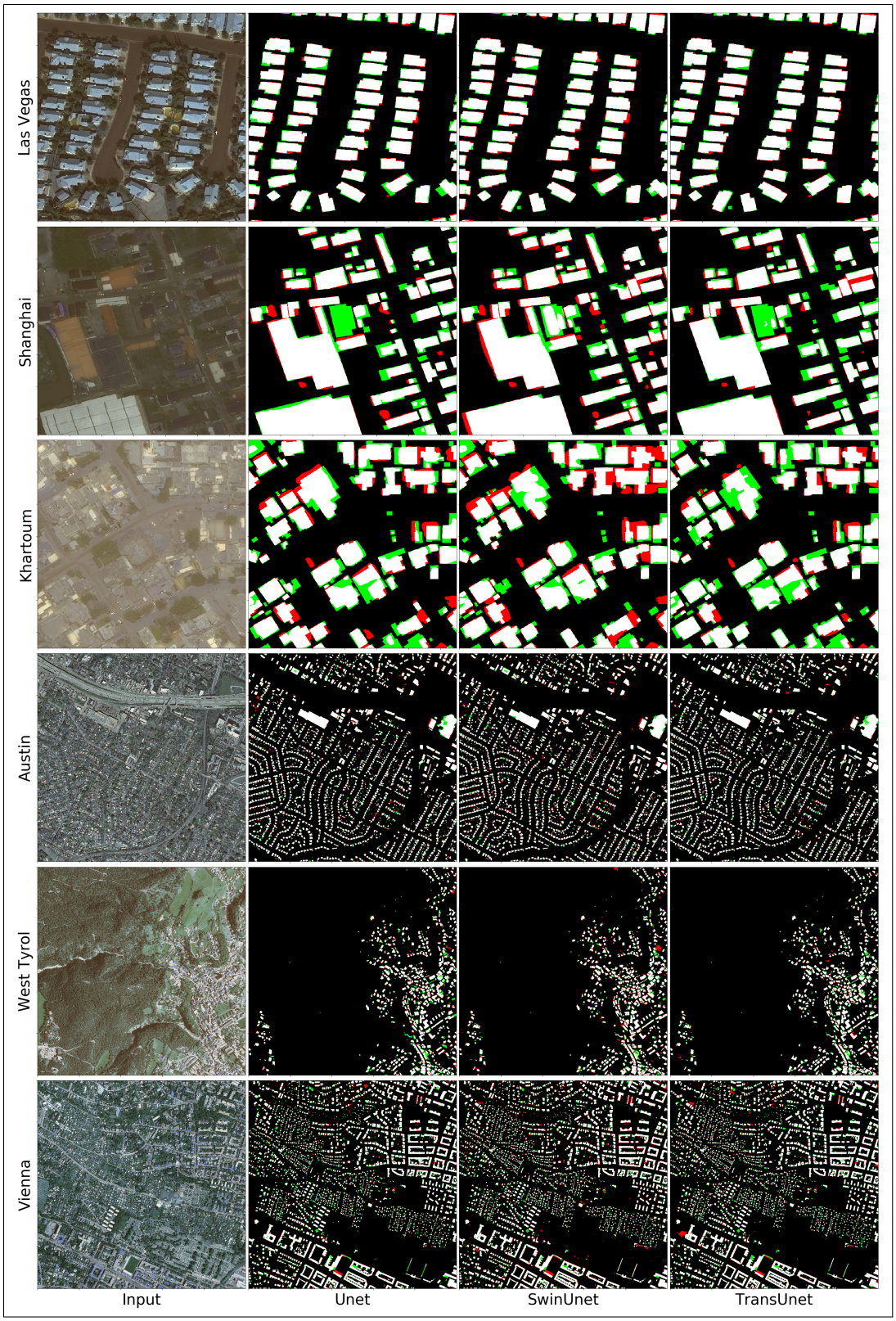

Figure 6. Examples of input images and mask outputs. Each row is a test image from a different city, with the first three from DeepGlobe and the last three from Inria. The first column contains the input image, the next three columns contain the predictions from the Unet, SwinUnet, and TransUnet respectively. The difference with the ground truth is highlighted, with green showing missed building pixels and red denoting false alarm building pixels.

References

[1] Hu Cao, Yueyue Wang, Joy Chen, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, and Manning Wang. Swin-unet: Unet-like pure transformer for medical image segmentation. *arXiv preprint arXiv:2105.05537*, 2021.

[2] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.