Comparing Transformers and CNNs on the SpaceNet Flood Detection Challenge

Adrian Stoll
Stanford University
adrs@stanford.edu

Naijing Guo
Stanford University
njguo@stanford.edu

Nenad Bozinovic
Stanford University
nesa@stanford.edu

Abstract

This project explores the SpaceNet8 Challenge, which aims to detect floods caused by hurricanes and heavy rains. We compared a variety of transformer and CNN segmentation architectures. We found that large pre-trained Segformer models had better performance than Resnet and U-net based models despite consuming more computational resources. The highest IoU for building detection was 61% for Segformer, which indicated that attention is better suited for detecting building-footprints than convolutions. We noticed that flooded road detection was particularly hard with highest IoU of 40%. We observed pre-training on ImageNet and Cityscapes datasets provided a moderate improvement compared to pre-training on the ADE20k dataset and a significant improvement compared to training a model from scratch.

1. Introduction

Floods are among the most devastating natural disasters, causing significant damage to buildings, infrastructure, and human life, as well as exacerbating environmental problems such as erosion and land degradation. Remote sensing, specifically satellite imagery analysis, has become a vital tool for flood detection and monitoring, providing fast initial mapping of flooded regions and helping in post-disaster assessments.

The SpaceNet8 Challenge [7] aims at developing open-source computer vision and machine learning methods for remote sensing data in the context of floods caused by hurricanes and heavy rains. The goal of this challenge is to rapidly develop maps and analyze the scale of destruction to better direct resources and first responders in real-world disaster scenarios by leveraging the existing repository of datasets and algorithms from SpaceNet Challenges 1-7 and expanding to multiclass feature extraction and characterization. We use the SpaceNet 8 dataset which has two sets of images, namely pre-event images and post-event images. We have two independently trained models. The first model is the foundational features network which takes the pre-event images as inputs and segments buildings and roads. This network returns two outputs with the same size as the original image and different number of channels - road speed that has 8 channels and building that has 1 channel. The second model is the flood network that takes both pre and post-event images as input and outputs predictions of flood status. The outputs have the same spatial dimensions as the the input image with 4 channels: flooded building, non-flooded building, flooded road, and non-flooded road. We investigated a few model options including transformer based and U-Net based models.

2. Related Work

Works related to automated detection and identification of buildings in off-nadir satellite imagery can be generally divided into three categories - deep learning based approach, Object-Based Image Analysis (OBIA), and hybrid approaches. In this section, we discuss their strengths and weaknesses.

2.1. Deep Learning Based Approach

2.1.1 Deeplab Approach

The Deeplab model [2] uses a few novel approaches to address the issues in semantic image segmentation. It takes advantage of upsampled filters - also known as 'atrous convolution' - to better control the resolution and incorporate larger context without increasing the number of parameters or the amount of computation. It also uses fully connected Conditional Random Field (CRF) to improve localization performance. And overall this approach has improved performance in delineating object boundaries, but has rough final segmentation results due to the lacks of considering low-level feature images with rich detailed information [17].

2.1.2 U-Net Based Approaches

U-Net is an encoder-decoder convolutional neural network that is a widely-used for semantic segmentation [12]. The network was first developed for application in biomedical image segmentation and was later extended to various fields

including remote sensing analyses. Its inherent strengths lie in its ability to effectively learn and capture local information through the combination of encoder and decoder architectures. However, in general U-Net struggles to grasp global contextual information, which can limit its performance [14].

2.2. Object-Based Image Analysis

Object-Based Image Analysis, like U-Net, was also originated from applications in biological image scans and later evolved into Geographic Object-Based Image Analysis that is used on remote sensing [1]. OBIA involves segmenting images into meaningful homogeneous regions or objects, which are further analyzed by feature and context extraction. This approach can better handle multiscale and heterogeneity problems, but requires extensive domain knowledge and a finely-tuned thresholding mechanism [5].

2.3. Hybrid Approaches

A hybrid approach combining CNNs and OBIA was proposed in the research area of ecology to better monitor the changes in vegetation cover on very high-resolution images [4]. The CNN extracts feature maps, which are fed into an OBIA module to refine the segmentation results. This approach was proposed to resolve the issue that the model performance could be affected by the spatial resolution of the orthoimages utilized, which presents in both CNN and OBIA based models. However, this model requires numerous high-quality high-resolution images, which are not always accessible in various fields of applications.

2.4. State-of-the-Art: SAM

The Segment Anything Model (SAM) is a groundbreaking image segmentation model that is is built upon a MAE pre-trained Vision Transformer model. It is designed to segment objects in images using zero-shot learning, which means it can generalize across domains without additional training [11]. We will discuss the use of SAM in our project in a later section.

3. Dataset

We use SpaceNet8 dataset described in detail here [7]. Data has been acquired using Maxar's Earth observation satellites (data under creative common license). For SpaceNet8 dataset, two distinct areas of interest (AOIs) were selected that were flooded in 2021 (Germany and Louisiana). Data consists of pansharpened RGB (0.3-0.8m resolution) that are routinely published through Maxar Open Data Program.

Data was pre-labeled by analysts for both the pre- and post-event images. A road segment is considered flooded if it is visibly covered with water or rubble in the post-event imagery. If a building is assigned the flooded attribute, it means that there is flood water in its immediate proximity which is a proxy for probable flooding (since it is not possible from satellite imagery to determine whether a building is actually flooded inside). In the event that cloud cover obscured features in the post-event image, as is often the case in flooding disasters, analysts labeled only visible areas and not make any assumptions where features could not be seen (Figure 1).

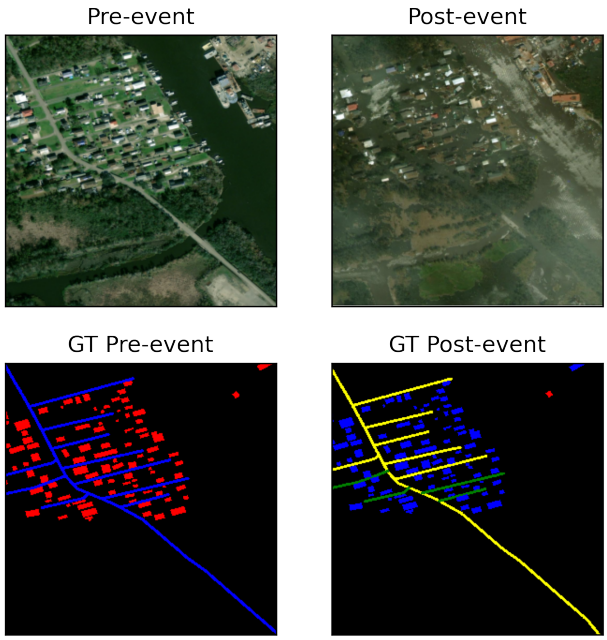

Figure 1. Examples of raw images pre- and post-event (top row) and respective ground truth segmenation masks (bottom row). Colors indicate classes (buildings, road, flooded buildings and flooded roads). In this example blue and yellow colors refer to flooded buildings and roads. For pre-event labels there is 1 building class and 8 road classes denoting speed (10 mph per class). For post event there are 4 classes (non-flooded building, flooded building, flooded road, non-flooded road).

Pre- and post-event images do not overlap perfectly but are close enough for our purpose. Masks are build from GeoJSON data, where, for example, only vertices of the building polygon mask are stored, vs full mask image (more details here).

At our disposal we had 801 labeled RGB images (1300x1300) and corresponding masks (Fig. 1). The focus of this project was to try different models and we have opted against hyper-parameter optimization, hence we had no validation dataset, and kept 122 images (15%) for testing. The images were not normalized or otherwise prepossessed before being fed into the models. All models were trained and evaluated using the same training-test split.

4. Pipeline

We have developed a streamlined pipeline (training and evaluation) that enables easy experimentation. The result of each *run* is a metrics file containing model name, parameter count, peak GPU memory usage, batch size, learning rate, per-epoch training and validation loss, as well as all Intersection-of-Union (IoU) metrics for each segmentation class and a best validation-loss model checkpoint.

Each model was trained on the entire training set for typically 10 epochs (45 min on A6000) after which we would typically see validation loss convergence and overfitting on the training dataset. Each training run used the Adam optimizer [10] with a learning rate of 0.0001 and cross-entropy as the loss function. We used the maximum possible batch size that would fit in GPU memory.

For evaluation, each model ran inference on all of the validation images to produces segmentation masks yielding per-pixel-precision, recall, and F1 score, and ultimately Intersection-over-Union (IoU) percentage between prediction for each class (e.g. non-flooded building, flooded building) and its labeled masks.

In the interest of time, we opted to modify the baseline pipeline provided by the Spacenet challenge organizers. The pipeline is notably different from most other pipelines as it uses two models: one for foundation feature detection (roads and buildings), and another so-called Siamese, encoder for flood detection. The architecture can be seen in Figure 2. The foundation features network passes the images thought an encoder-decoder backbone followed by two sets of convolutions layers - one outputting building and the other road network predictions. The flood attribution network passes pre- and post-event images thought encoder-decoder backbones with shared weights (hence the name Siamese network) to extra image features. The flood network then concatenates these features together along the channel dimension and passes the features thought a sequence of convolution layers to output the flooded/not-flooded road/building predictions.

5. Methods

We experimented with various backbones for the flood network and Siamese foundation features network to compare the performance and characteristics of convolutional U-net and transformer model families. The families of models we evaluated include Segformer [19], DenseNet [6], and EfficientNet [16]. The baseline the models were compared against is the reference SpaceNet pipeline [15] with a Resnet34 backbone released by the competition organizers. We also experimented fine-tuning backbone models with pre-trained weights from ImageNet [13], ADE20k [20], and Cityscapes [3] datasets as well as training from scratch.

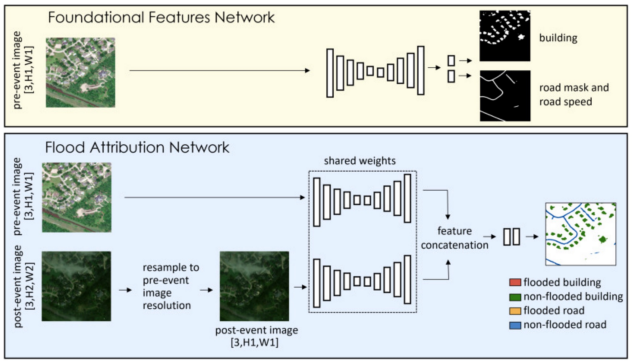

Figure 2. The foundation features network and flood attribution networks. The pipeline design is modular and allows different backbone models to be swapped in while maintaining the same data-loading, training, and evaluation code.

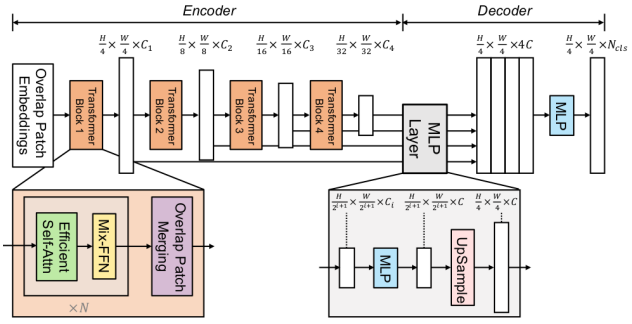

Figure 3. The Segformer model architecture has a transformer encoder and MLP decoder. The hierarchical structure is intended to capture multiple-scale features. Because the output reduces the Spatial dimension by a factor of 4, we add an upscaling layer to preserve image dimensions.

5.1. SegFormer

We experiment with increasingly large Segformer backbones to compare transformer based architectures with the baseline of CNNs. Segformer uses a hierarchical transformer encoder combining local and global attention to achive start-of-the-art performance on image segmentation tasks. Unlike ViT, Segformer does not use positional encoding which aids transfer learning between datasets with different image resolutions. This is important because the SpaceNet8 images have different resolutions than the images from the ImageNet, ADE20k, and Cityscale datasets used for pre-training.

We used Segformer b0, b1, and b2 models from the Huggingface transformers library [18]. The difference between these variants is the number of parameters. We did not test the larger b3, b4, and b5 variants because they do not fit in the A6000 GPU's memory. Segformer reduces the spatial dimension by a factor of 4. We use nearest neighbor interpolation (link) to upscale the segformer output to match the

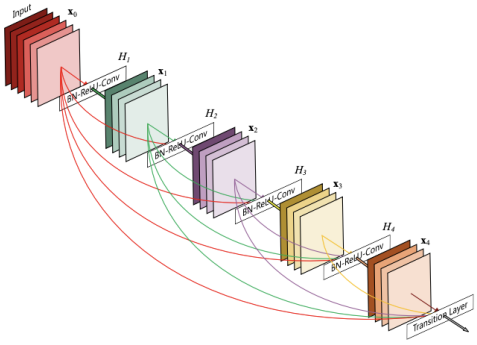

Figure 4. DenseNet with 5 layers with expansion of 4

input spatial resolution. The models weights were initialized with pre-trained weights released by NVidia and each model was fine-tuned without freezing any parameters.

5.2. DenseNet

We experimented with DenseNet as one of the encoder structure for the U-net models. In traditional CNNs, the output of a layer is connected to the next layer after applying a series of operations, such as convolution, pooling, batch normalization, and an activation function. However, DenseNets differ in that they concatenate the output feature maps of a layer with the incoming feature maps, rather than summing them. To accommodate feature maps of different dimensions, DenseNets are divided into DenseBlocks, where the dimensions of the feature maps remain constant within each block, but the number of filters may vary towardsdatascience.com. To connect these DenseBlocks, transition layers are used, which handle operations such as downsampling and batch normalization. The advantage of DenseNet as stated in the original paper was that, as of the performance on ImageNet, it has similar performance as ResNet, but uses fewer than half as many parameters [6]. We implemented DenseNet 121 and DenseNet 161 with weights pre-trained on ImageNet. The main difference between the 121 and 161 model is the size - DenseNet 121 is about 13MB in size while DenseNet 161 is about 100MB [8].

5.3. EfficientNet

Another encoder structure we experimented with was EfficientNet. The main difference between EfficientNet and traditional convolutional neural network is that EfficientNet uniformly scales all dimensions of depth/width/resolution using a compound coefficient. The advantage of EfficientNet is that it can generally achieve good accuracy on image classification tasks, such as ImageNet and transfer learning tasks, while requiring significantly fewer floating-point operations (FLOPs) for inference [9]. EfficientNet consists of

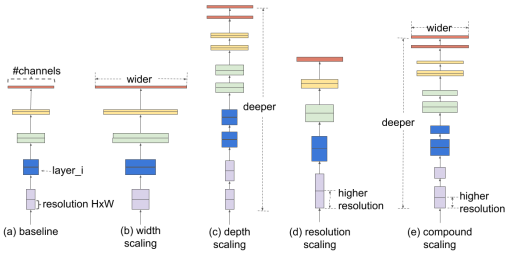

Figure 5. (a) Baseline network, (b)-(d) Conventional scaling, (e) Compound scaling in EfficientNet

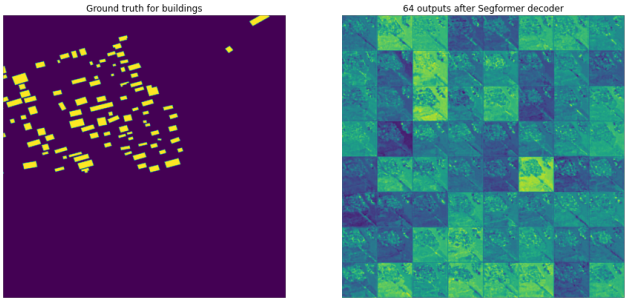

Figure 6. The left side shows the ground true segmentation mask for a group of buildings. The right sides shows an the Segformer decoder output for the image. The 64 output channels are arranged in a 8x8 grid of image blocks.

a series of models (B0 to B7) that represent a good balance between efficiency and accuracy across different scales. We experimented with B2 and B4 models.

6. Experiments

6.1. SegFormer head

The Segformer architecture (encoder-bottleneck-decoder) pre-trained on ImageNet produces a $(64, \frac{N}{4}, \frac{N}{4})$ mask and require a segmentation head to reduce the number of channels to 1 or 8 for the foundation features model and 4 for the flood attribution model. To understand the model more, we have visualized the decoder output Fig. 6. We experimented with three segmentation heads (single 1x1 convolution, single 3x3 convolution, and double 3x3 convolution). All work at this point was one using Segformer-b0. Loss and IoUs for all trials was similar; IoUs of 56.7% (1x1 conv), 56.6% (double 3x3), 57.1% (3x3 conv) and we settled on a single 3x3 convolution layer. The training loss curves are shown in Fig. 7.

6.2. Pre-training

We experimented with fine tuning models initialized from pre-trained weights. We observed a significant improvement in IoU for both Resnet34 and Segformer_b0

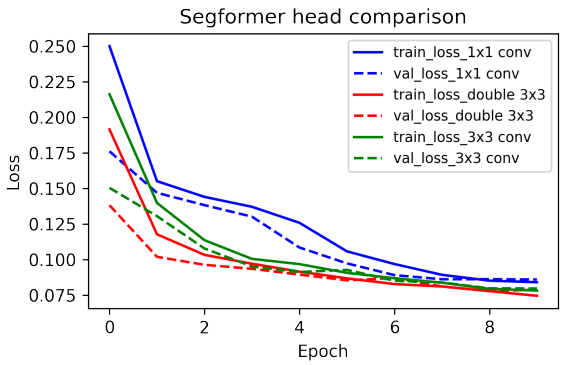

Figure 7. Loss for different Segformer segmentation heads were similar and we settled on a single 3x3 convolution layer.

flood and foundation feature networks (e.g. 0.56 IoU vs 0.36 IoU for the Resnet34 foundation features model). Additionally we compared pre-training the Segformer model on ImageNet, ADE20k, and Cityscapes datasets. The ImageNet dataset consists of images scraped from the internet of objects like animals, food, cars, etc [13]. The ADE-20k dataset consists of annotated images with scenes of kitchens, bedrooms, stores, restaurants, etc [20]. The Cityscapes dataset consists of annotated images of streets from 50 different cities [3]. We observe pre-training with ImageNet and Cityscapes yields an improvement over pre-training with ADE20k for the Foundation Features network and Flood network (See tables 2 and 2). These results are expected because the outdoor images in the Cityscapes dataset are more similar to the SpaceNet8 dataset than the mostly indoor images from the ADE20k dataset.

An interesting observation is al the models are biased (higher IoUs) towards assigning not-flooded labels to buildings and roads. The worst case is the model fine-tuned from ADE20k weights which has zero IoU for flooded buildings and roads. This is due to the class imbalance in the SpaceNet8 dataset. Only 13% of buildings and 15% of roads are flooded. The flood network is trained using the cross-entropy loss function which does not adjust for the data imbalance. If we had more time we could use dice loss which is designed for highly unbalanced segmentation tasks. Even with the poor choice of loss function, the best Segformer model is still able to achieve reasonable IoUs for flooded roads and buildings.

6.3. Compute Resource Analysis

All of our computations were done on the Paperspace platform initially using P6000 GPUs, and later A6000. In order to understand hardware limitations we studied some of the GPU components across different models, chiefly: space and time complexity in the form of memory consumption and epoch duration respectively. As expected we

Model (Pre-training)	Building IoU	Road IoU
Resnet34 (None)	0.36	0.35
Resnet34 (Imagenet-1k)	0.56	**0.48**
Segformer_b0 (None)	0.28	0.29
Segformer_b0 (Imagenet-1k)	**0.59**	0.47
Segformer_b0 (ADE20k)	0.54	0.44
Segformer_b0 (Cityscapes)	0.58	0.47

Table 1. Comparison of pre-trained vs not pre-trained Foundation Features networks. The best IoUs for Building and Road segment are in bold. Pre-training provides a 0.13 to 0.31 increase in IoU.

Model (Pre-training)	NF Bldg	F Bldg	NF Road	F Road
Resnet (None)	0.28	0.03	0.10	0.00
Resnet (Imagenet)	0.46	0.12	0.30	0.18
Segformer (None)	0.22	0.01	0.13	0.00
Segformer (Imagenet)	0.58	0.38	**0.43**	**0.37**
Segformer (ADE20k)	0.46	0.0	0.31	0.03
Segformer (Cityscapes)	**0.61**	**0.40**	0.41	0.34

Table 2. Comparison of pre-trained vs not pre-trained Flood networks. Pre-training the Segformer model on the Cityscapes dataset provides a 0.39 increase in IoU for both Not-Flooded Building (NF Bldg) and Flooded Building (F Bldg) classes and Pre-training on ImageNet provides similar increase for Road segmentation.

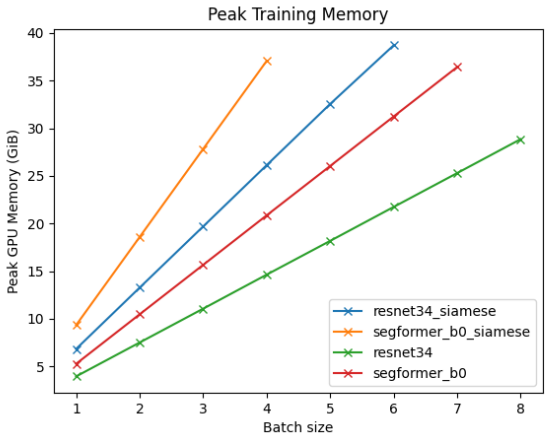

Figure 8. Peak GPU memory consumption for different models and different batch sizes.

observed large differences among models.

We measured the peak GPU memory consumed by each model for various batch sizes because GPU memory constrained the size of the Segformer models we could train on the A6000 GPU. The y-intercept of the memory vs batch-size best fit line is the memory from model weights, their gradients, and overhead. The slope represents the memory for training examples, activations, and gradients of ac-

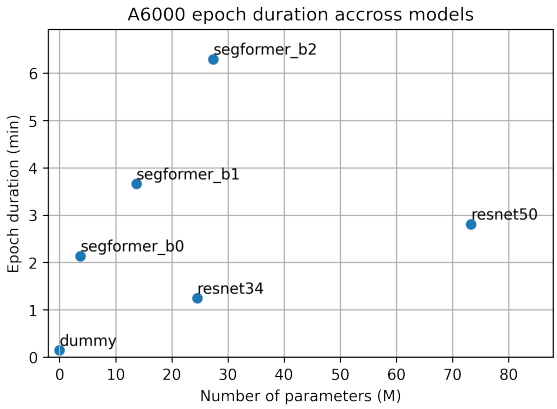

Figure 9. Total epoch duration on the A6000 GPU compared with the number of parameters for each model. As a baseline we trained a dummy model consisting of a single 1x1 conv layer projecting from the input to output dimension.

tivations. The Siamese networks consume almost twice as much memory for a given batch sizes because they operate on two images per training-example vs a single image per training-example for the Foundation features network. Additionally we observe Segformer_b0 consumes more memory that Resnet34 despite having fewer parameters which can be described by fundamentally different calculation done by attention vs convolution. Indeed, another difference between CNN vs Transformed based models can be seen in epoch duration (Figure 9). Segformer (b0, b1, b2) all have longer epochs even though they have fewer parameters. This is not surprising considering attention has quadratic complexity in comparison to CNN linear. (Transformer time complexity is $O(n^2d + nd^2)$ where n is the number of image patches and d is the size of the vector representations. (n,d) are dimensions of query, key, and value matrices in self-attention).

We also noticed one particular quirk regarding data loading from hard drive into CPU RAM, namely, the difference between using shared network drive and local hard drive. For quite some time we have employed a shared network drive for all of our data storage to facilitate working on multiple instances. Typical epoch would take 4.5 minutes regardless of the model we were testing. This was a surprising result and we initially assumed that the bottleneck was in the amount of time it takes to transfer data from CPU to GPU memory. The truth was different, and follow up experiment of loading data from a local hard drive cut the training time of Resnet34 3-fold. A stark 6.5X difference in data loading speed from shared network drive (4 minutes) vs local hard drive (40 seconds) was causing data loading to be computational bottleneck for all but Segformer_b2 model (considering CPU data loading and GPU computation happen in parallel). It is a cautionary tale that we will keep in mind when training future models.

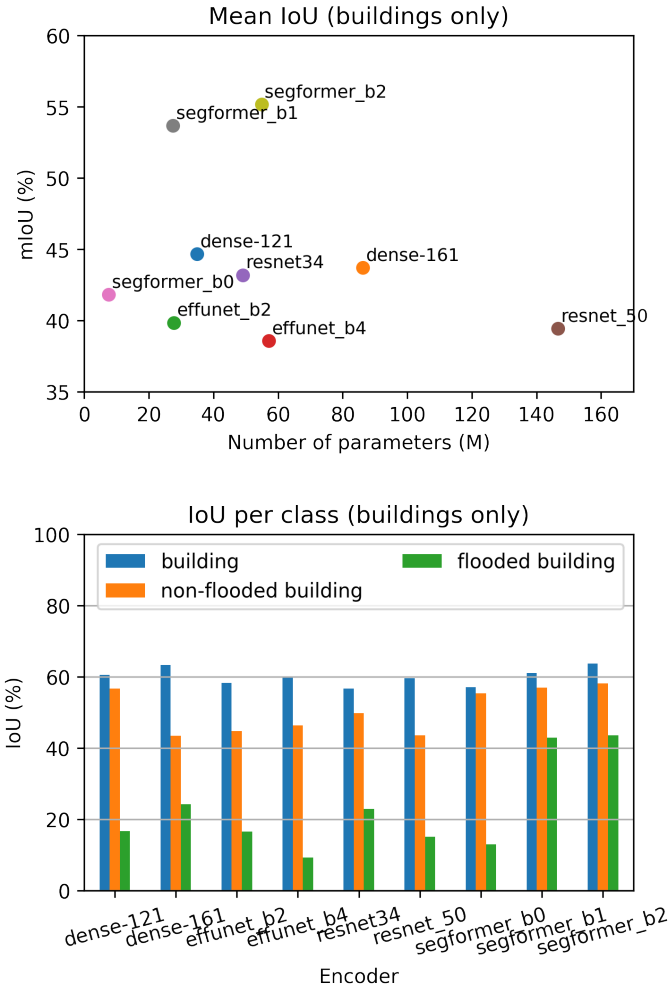

Figure 10. IoU comparison between models for building detection only. Segformer models outperform CNN based one, showing that attention is better suited for detecting building and road-like polygonal objects.

6.4. Model comparison

We finally compare IoUs and number of parameters across all models (Figure 10). While most models performed similarly when detecting buildings in pre-event images, Segformer_b1 and b2 emerged as much better option when detecting flooded buildings (40%) vs others (Table 2). Qualitatively Segformer is noticeably better at detecting polygonal shapes with sharp edges vs blob-like structures detected by CNN based models. This can be explained as another attention advantage over CNN. For qualitative comparison refer to Figures 12-17 in Appendix.

7. Discussion

We have used early-stopping for all models (roughly 10 epochs) where validation loss would plateau. We did notice overfitting on training dataset (> 10 epochs) that could be explained with the fact that all models have large amount of parameters (in the millions) while there are only hundreds of training images (adding more training data would have helped of course). However, since we didn't observe directly overfitting on a validation dataset we consider that all models generalized well.

Road detection seems to have been much harder then building detection. Qualitatively, road have been detected especially in the pre-event images. It is possible that IoU might not be the best metric when it comes to road detection considering it's elongated shape. It is possible that misalignment of the pre- and post-images could have caused whole roads to be missed. We admit improvement can be done to this end (SpaceNet3 dataset was particularly designed for read detection and can serve for further inspiration).

8. Conclusion/Future Work

In conclusion, we found that large pre-trained Segformer models outperformed Resnet and U-net based models. We hypothesize that the pre-trained models performed better because we only trained for a relatively small number of epochs on a small dataset of 700 images. With a larger dataset and more epochs, the advantage from pre-training would shrink. Additionally, different convolution layers were tested for the segmentation head, with similar IoUs of 56.7% (1x1 conv), 56.6% (double 3x3), and 57.1% (3x3 conv), ultimately settling on a single 3x3 convolution layer. This project also compared memory consumption and epoch duration across different models, observing that SegFormer models consume more memory and have longer epochs despite having fewer parameters compared to ResNet34. We reasoned that attention has quadratic complexity, contributing to the longer epoch durations for SegFormer models. Another interesting observation was the impact of data loading from a network attached storage versus a local hard drive. Loading data from a local hard drive took only 40 seconds, while loading from a shared drive took 4.5 minutes, resulting in a 6x speedup when using a local hard drive. This difference in data loading time highlights the importance of data storage and access methods when working with computationally intensive models like SegFormer.

As for future work, we propose to implement several techniques to improve the segmentation accuracy of our deep learning model. First, we will normalize images and apply other pre-processing techniques to reduce variance and improve model convergence. Second, we will pre-train the model on additional datasets, such as building and road network datasets from previous SpaceNet challenges, to improve its ability to recognize different features in satellite images. Since we observed that different models could have different performance in specific segmentation task - for example, pre-training on ImageNet is better for road segmentation, while pre-training on Cityscapes is better for flooded building segmentation - we could also combine models into ensembles to improve the performance. Finally, we will change the loss function for the flood network from cross-entropy to one that accounts for class imbalances to improve accuracy in the segmentation of flooded buildings. In addition to the potential work we can do with our existing models, there are a few different model architectures that we can explore. We tried the state-of-art model Segment Anything (SAM) on our image. The model was able to segment the image neatly into different parts but the segmentation was too granulated and was not suitable for our application (11) - for example the trees on the field was segmented but a road was missed. This result is in fact not bad given that it is zero-shot. In the future we could try to fine-tune this model so that if focuses on the specific instances we want and gives better performance for our tasks.

Figure 11. Example of Segment Anything prediction with zero-shot learning on satellite images. Fine-tuning it could be a next promising step.

9. Contributions & Acknowledgements

We made the following contributions and utilized the following GitHub repositories:

- Adrian: Segformer model integration, peak-GPU memory measurements, pre-trained vs not pre-trained experiments.
- Naijing: EfficientNet and Densenet model integrations.
- Nenad: IoU vs number of parameters measurements, segment anything, Segformer head comparison, epoch runtime comparison.

- Repositories: SpaceNet8 baseline pipeline, Segformer model, EfficientNet model, Segment Anything, DenseNet, 5th place SpaceNet8 Challenge Entry.

10. References/Bibliography

References

[1] Thomas Blaschke. Object based image analysis for remote sensing. *ISPRS journal of photogrammetry and remote sensing*, 65(1):2–16, 2010. 2

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018. 1

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016. 3, 5

[4] Emilio Guirado, Javier Blanco-Sacristán, Emilio Rodríguez-Caballero, Siham Tabik, Domingo Alcaraz-Segura, Jaime Martínez-Valderrama, and Javier Cabello. Mask r-cnn and obia fusion improves the segmentation of scattered vegetation in very high-resolution optical sensors. *Sensors*, 21(1):320, 2021. 2

[5] Geoffrey J. Hay and Guillermo Castilla. Object-based image analysis: Strengths, weaknesses, opportunities and threats (swot). In *Proc. 1st Int. Conf. OBIA*, 2006. 2

[6] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2261–2269, 2017. 3, 4

[7] Ronny Hänsch, Jacob Arndt, Dalton Lunga, Matthew Gibb, Tyler Pedelose, Arnold Boedihardjo, Desiree Petrie, and Todd M. Bacastow. Spacenet 8 - the detection of flooded roads and buildings. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1471–1479, 2022. 1, 2

[8] Intel Corporation. Densenet. https://docs.openvino.ai/2021.4/omz_models_model_densenet_161.html, 2021. Accessed: 2023-06-08. 4

[9] Keras Team. Image classification with efficientnet fine-tuning. https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/, 2021. Accessed: 2023-06-08. 4

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 3

[11] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 2

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 1

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy,

Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 3, 5

[14] Maysam Shahedi, Anusha Devi, James D Dormer, and Baowei Fei. A study on u-net limitations in object localization and image segmentation. In *Society for Imaging Informatics in Medicine (SIIM), Virtual meeting*, 2020. 2

[15] SpaceNetChallenge. Spacenet8. https://github.com/SpaceNetChallenge/SpaceNet8, 2022. 3

[16] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*. PMLR, 2019. 3

[17] Y Wang, C Wang, H Wu, and P Chen. An improved deeplabv3+ semantic segmentation algorithm with multiple loss constraints. *PLoS One*, 17(1):e0261582, 2022. 1

[18] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. 3

[19] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers, 2021. 3

[20] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 3, 5

11. Appendix

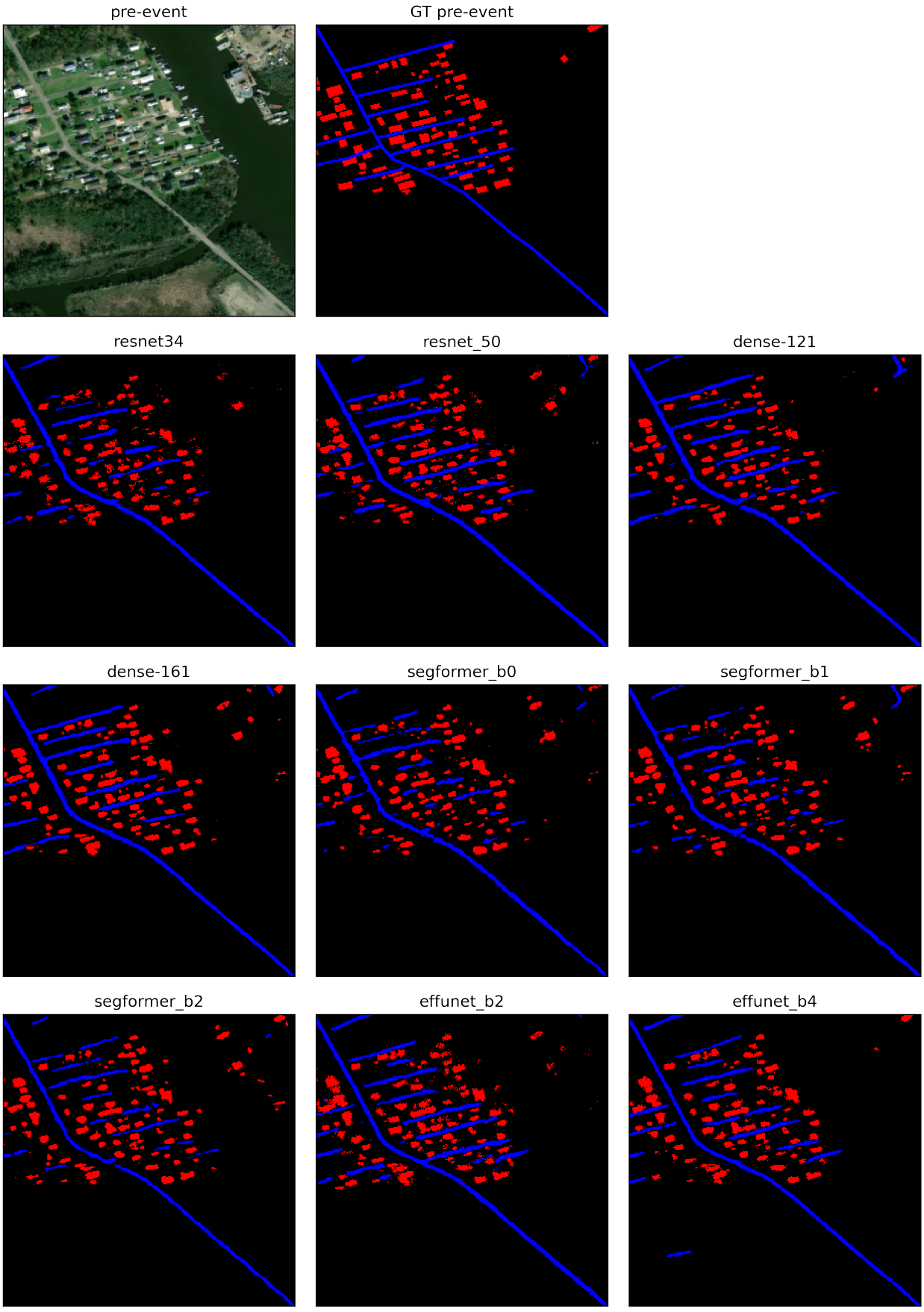

Figure 12. Predictions across Foundation network models.

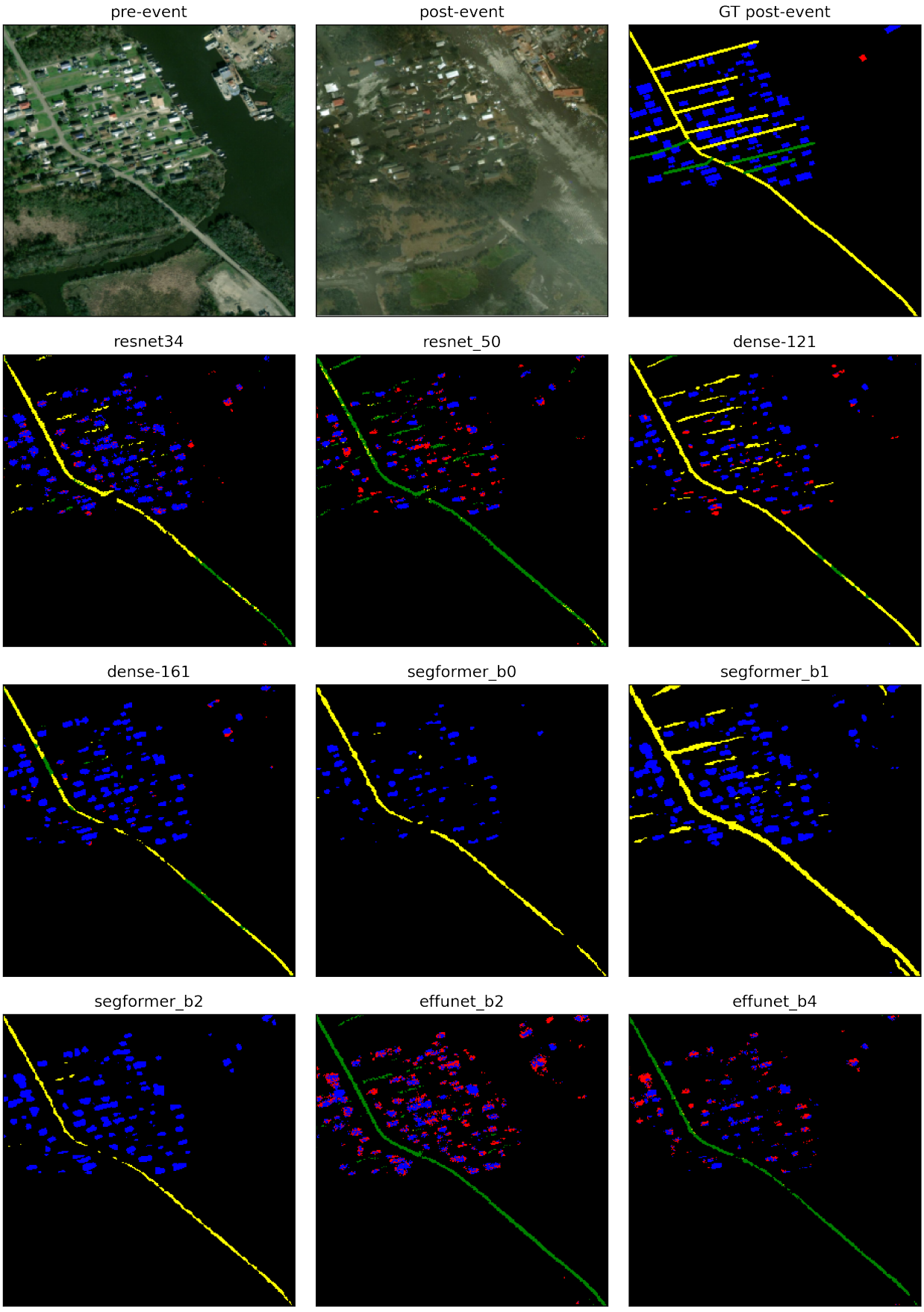

Figure 13. Predictions across Flood network models.

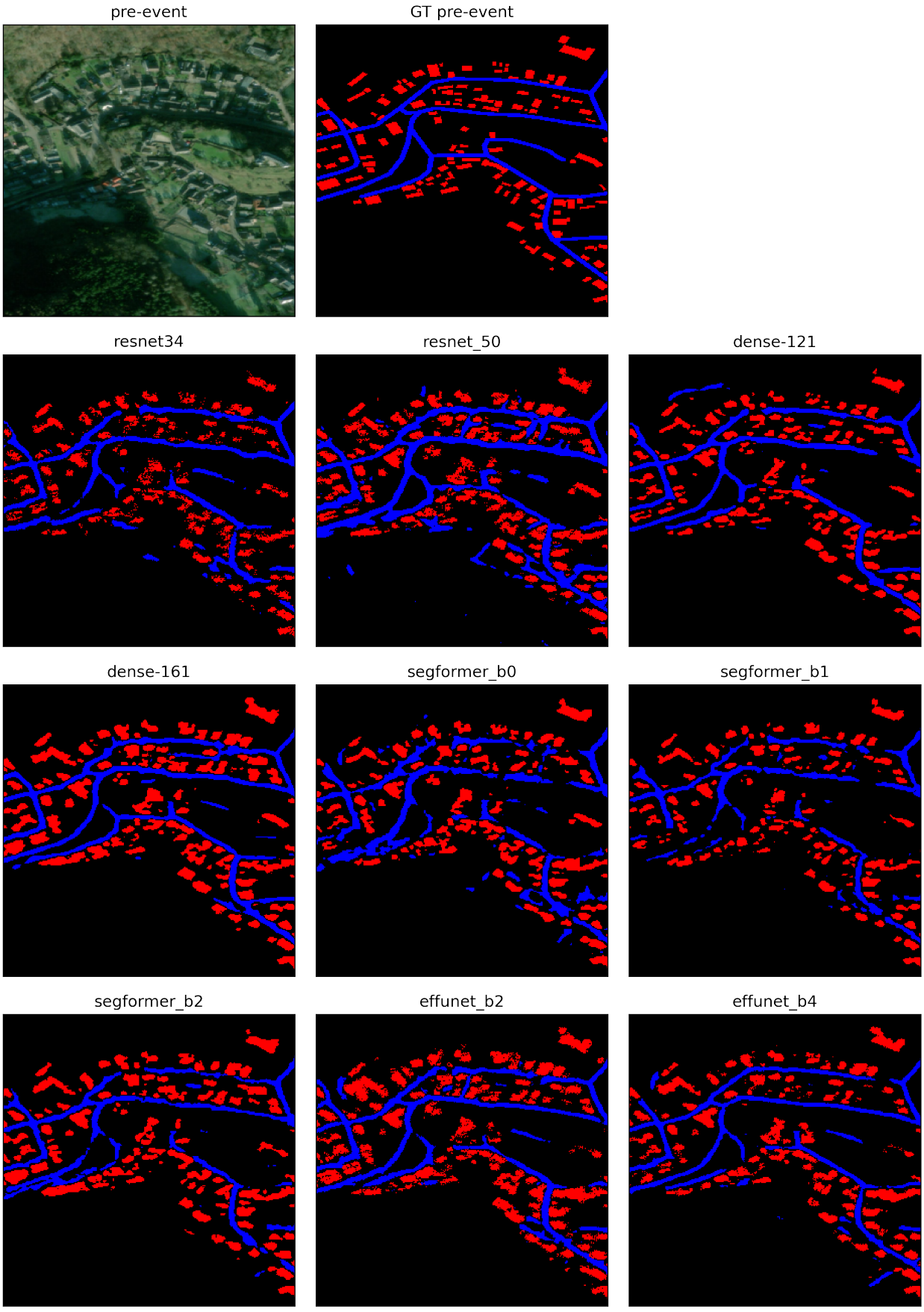

Figure 14. Predictions across Foundation network models.

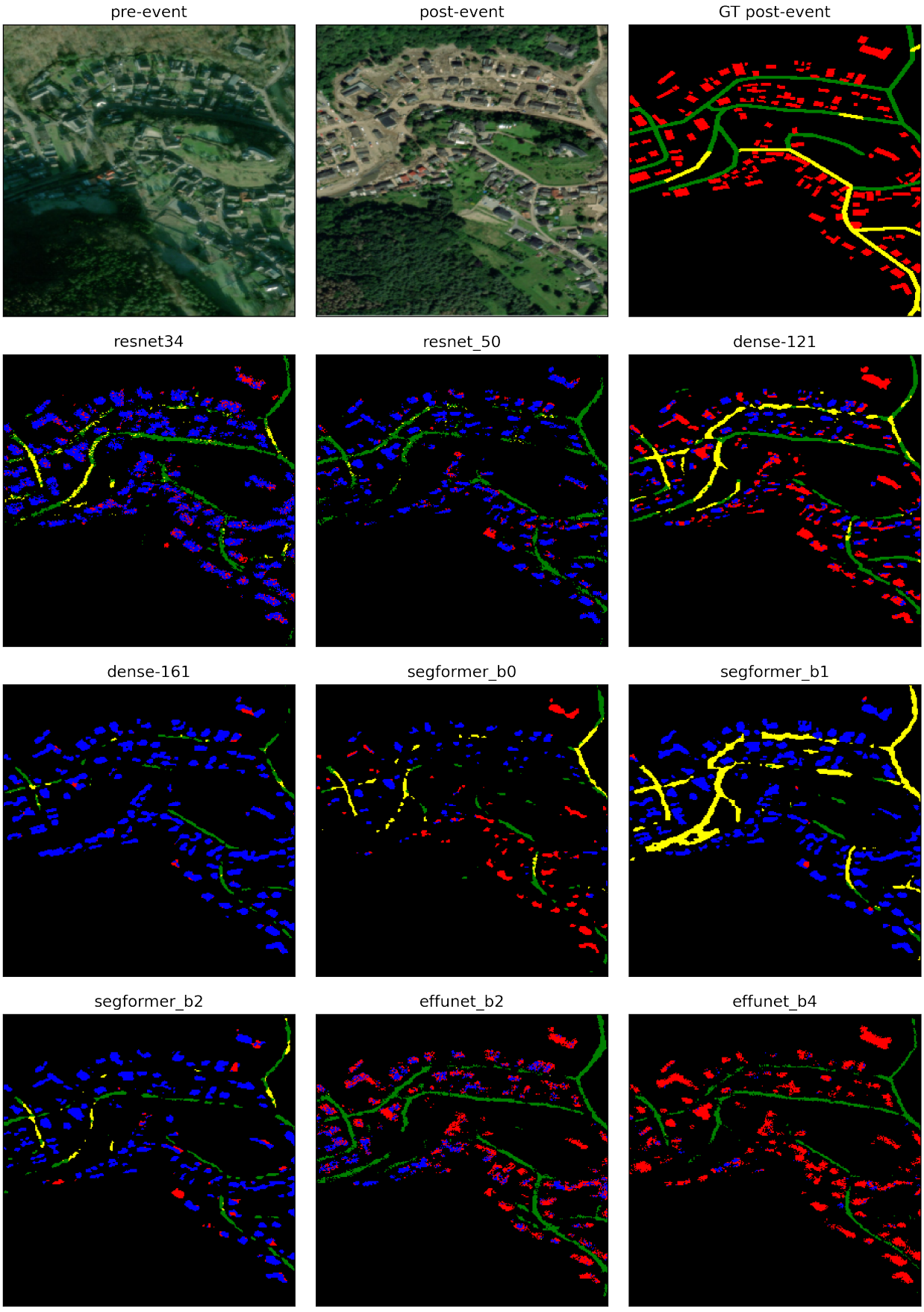

Figure 15. Predictions across Flood network models.

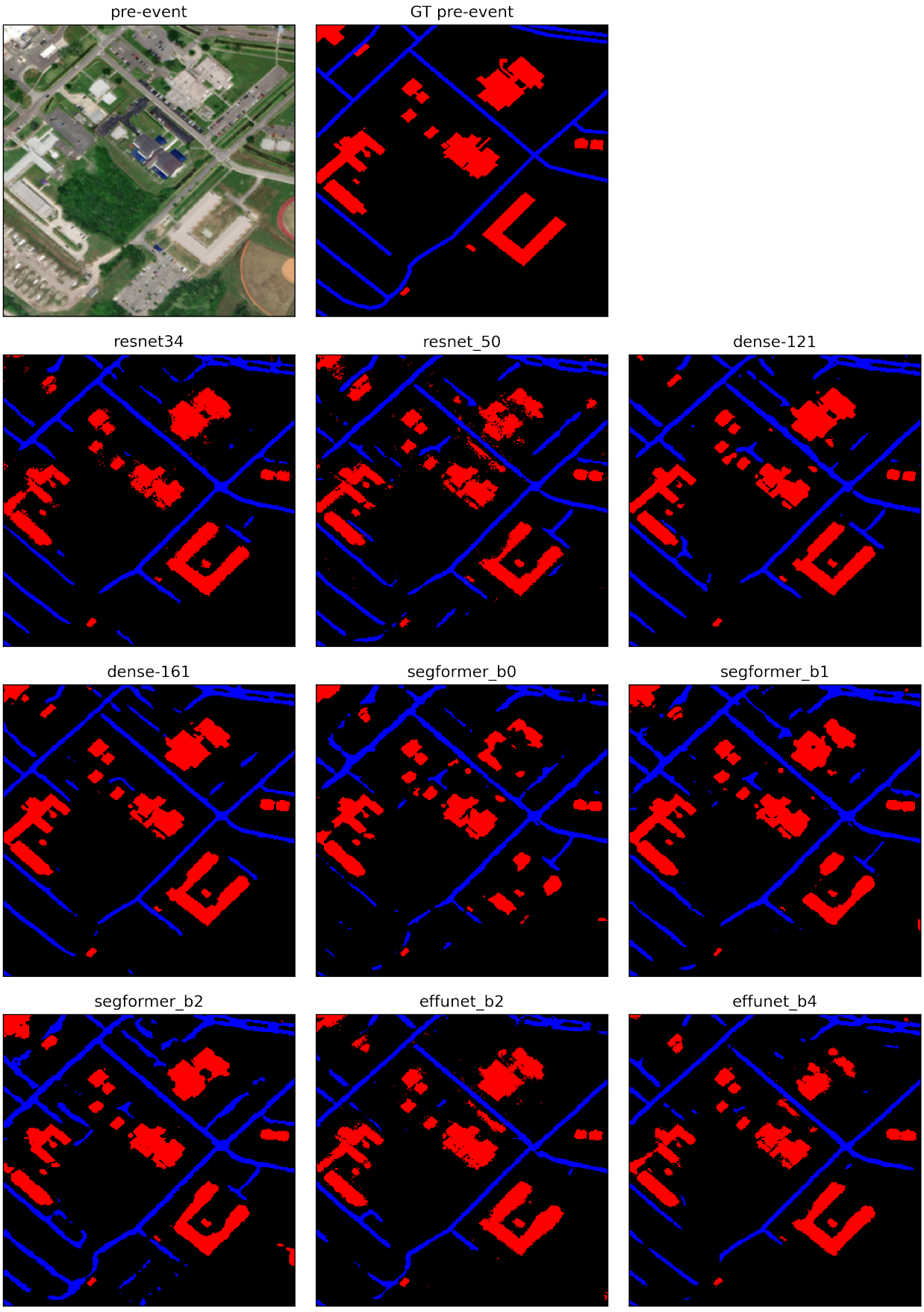

Figure 16. Predictions across Foundation network models.

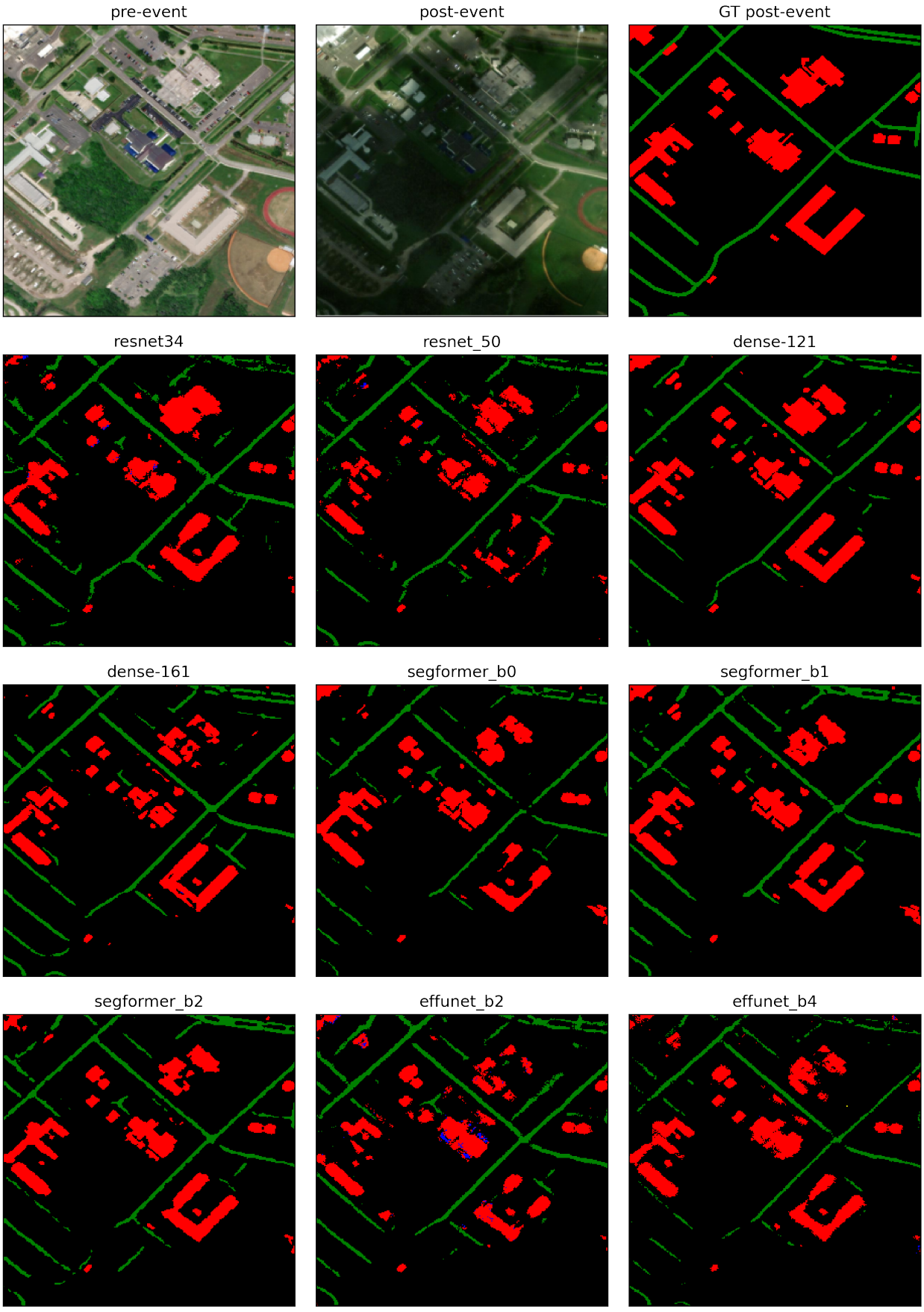

Figure 17. Predictions across Flood network models.