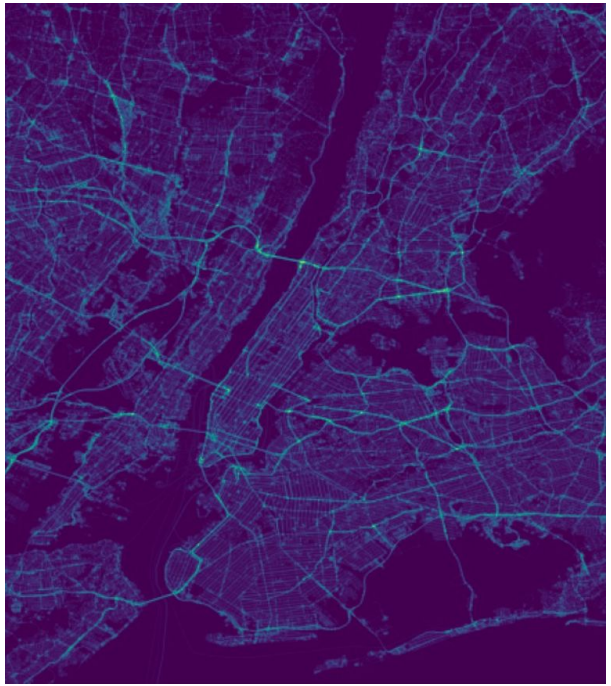
Traffic forecasting on traffic movie snippets

Nina Wiedemann and Martin Raubal

Chair of Geoinformation Engineering, ETH Zürich

Motivation

- Generalization to new cities in the *Extended Challenge*
- Advantages patch-based approach:
  - Reduce input dimensionality
  - Not required to compress all cities into a raster of the same size
  - Patch-wise processing yields multiple outputs per pixel



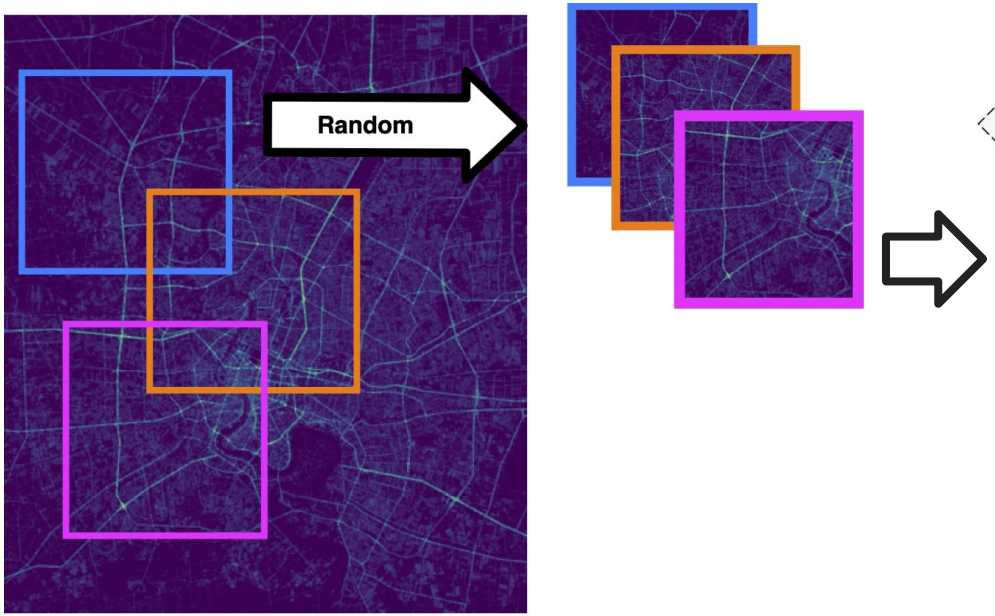
Related work

- Patch-based processing was proposed in multiple Computer Vision applications
  - Natural image segmentation [1]
  - Tumor segmentation [2, 3]
  - Brain image segmentation [4]
  - Tree bark classification [5]

- Usually, predictions are summarised by averaging or majority voting

[1] Zhang, Lei, et al. "An image segmentation framework based on patch segmentation fusion." *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 2. Ieee, 2006.
[2] Ghimire, Kanchan, Quan Chen, and Xue Feng. "Patch-based 3D UNet for head and neck tumor segmentation with an ensemble of conventional and dilated convolutions." *3D Head and Neck Tumor Segmentation in PET/CT Challenge*. Springer, Cham, 2020.
[3] Do, Nhu-Tai, et al. "Multi-Level Seg-Unet Model with Global and Patch-Based X-ray Images for Knee Bone Tumor Detection." *Diagnostics* 11.4 (2021): 691.
[4] Zhang, Wenlu, et al. "Deep convolutional neural networks for multi-modality isointense infant brain image segmentation." NeuroImage 108 (2015): 214-224.
[5] Misra, Debaleena, Carlos Crispim-Junior, and Laure Tougne. "Patch-based CNN evaluation for bark classification." European Conference on Computer Vision. Springer, Cham, 2020.

Training



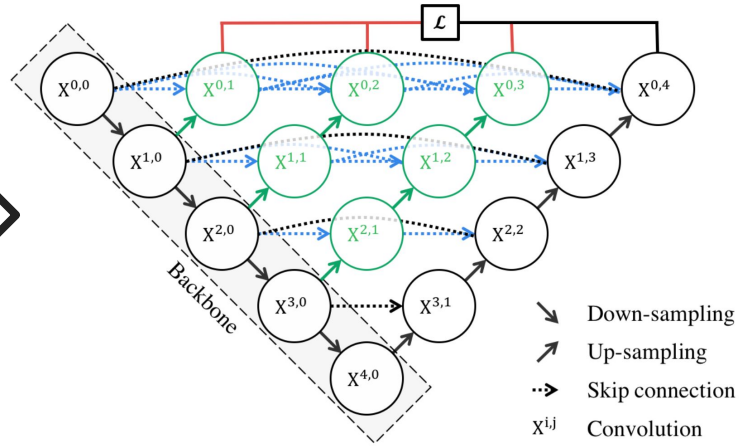


Image taken from:
https://www.arxiv-vanity.com/papers/1807.10165/

→ Extract patches randomly and pass through Unet++ [1]

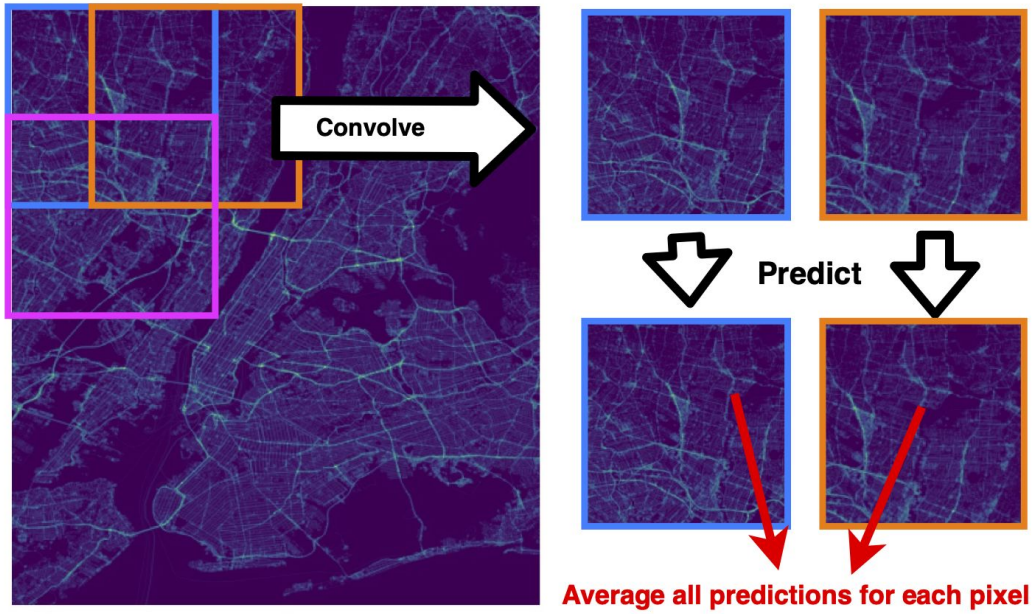
[1] Zhou, Zongwei, et al. "Unet++: A nested u-net architecture for medical image segmentation." *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, Cham, 2018. 3-11.

Split and merge at test time

1) Split new city into overlapping patches in a regular grid (move window with stride s)

2) Predict patches

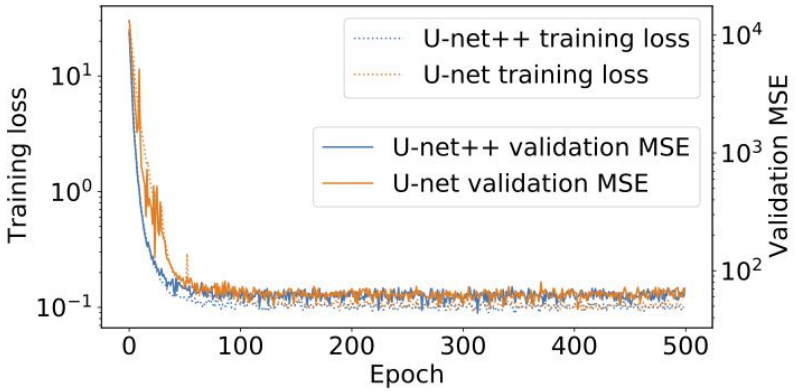
3) Average predictions per pixel



Results - Leaderboard

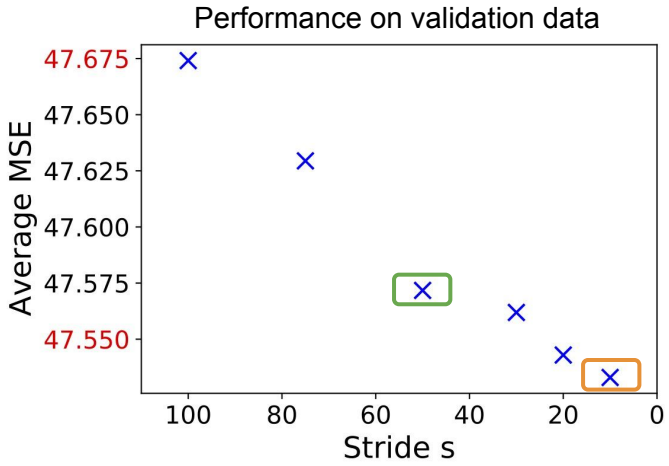
- Unet++ outperformed the standard U-Net architecture in our experiments
- Training on patches rather than whole cities significantly improves the result in the extended challenge
- Lowering the stride improves performance

Model	Patch size (d x d)	Stride	Score (MSE)
U-Net	495 x 436	-	64.2
Unet++	495 x 436	-	60.93
U-Net	100 x 100	50	61.32
Unet++	100 x 100	50	60.134
Unet++	60 x 60	30	60.44
Unet++	100 x 100	10	59.93



Results - Stride analysis

Model	Patch size (d x d)	Stride	Score (MSE)
U-Net	495 x 436	-	64.2
Unet++	495 x 436	-	60.93
U-Net	100 x 100	50	61.32
Unet++	100 x 100	50	60.134
Unet++	60 x 60	30	60.44
Unet++	100 x 100	10	59.93



Only weak decrease of the MSE when convolving the test data with more overlapping patches (low “stride”)

→ The main advantage comes from the patch-wise processing itself, not from the ensemble-type averaging

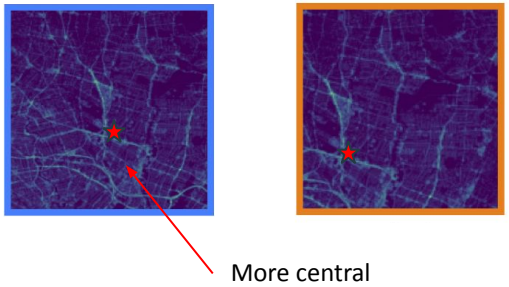
Results - Analysis of prediction averaging

Is averaging of the predictions the best aggregation method?

Hypothesis: The more central a pixel is in the patch, the better the prediction.

Three methods tested to aggregate predictions:

- Simple average over patch-wise prediction
- Use the pixel prediction from the **one** patch where the pixel is most central
- Weighted average with weights ~ centrality of the pixel in the patch



→ Averaging works best (on validation data)

Average	Most central	Weighted by pixel centrality
43.81	43.98	43.85

Conclusion

- Patch-wise processing has multiple advantages:
  - Better generalization
  - Faster training
  - Ensemble-like output averaging
- Compressing all cities into a predefined raster is probably not necessary or even disadvantagous
- Combine patch-based approach with better model architectures might achieve best performance
- Further work:
  - Adapt loss functions (e.g. masking)
  - Test methods developed for *sparse* image data

Acknowledgements

Thanks to Martin Raubal and my colleagues at MIE Lab for their support!

- Traffic4cast code: https://github.com/NinaWie/NeurIPS2021-traffic4cast
- arXiv paper: https://arxiv.org/abs/2110.14383
- Website: http://mie-lab.ethz.ch
- Our GitHub page: https://github.com/mie-lab