
Solving Traffic4Cast Competition with U-Net and Temporal Domain Adaptation

Vsevolod Konyakhin
ITMO University
Saint Petersburg, Russia
sevakonyakhin@gmail.com

Nina Lukashina
JetBrains Research
Saint Petersburg, Russia
nina.lukashina@jetbrains.com

Aleksei Shpilman
JetBrains Research
HSE University
Saint Petersburg, Russia
alexey@shpilman.com

Abstract

In this technical report, we present our solution to the Traffic4Cast 2021 Core Challenge, in which participants were asked to develop algorithms for predicting a traffic state 60 minutes ahead, based on the information from the previous hour, in 4 different cities. In contrast to the previously held competitions, this year’s challenge focuses on the temporal domain shift in traffic due to COVID-19 pandemic. Following the past success of U-Net, we utilize it for predicting future traffic maps. Additionally, we explore the usage of pre-trained encoders such as DenseNet and EfficientNet and employ multiple domain adaptation techniques to fight the domain shift. Our solution has ranked third in the final competition. Our code is available at <https://github.com/jbr-ai-labs/traffic4cast-2021>.

1 Introduction

The Traffic4Cast Competition, organized over the last three years by the Institute of Advanced Research in Artificial Intelligence (IARAI) and the data-provider company, HERE technologies,¹ challenges its participants to design algorithms that would predict future traffic states based on the prior traffic information. Specifically, the Traffic4Cast 2019 Challenge [1] focuses on forecasting the traffic state 15 minutes ahead based on the dynamic information from the previous hour, while the prediction window is increased to up to an hour in the Traffic4Cast 2020 Challenge [2].

This year’s edition of the competition, Traffic4Cast 2021, has introduced two competition tracks:

- Core Challenge, in which participants are tasked to fight a temporal domain shift that appears due to COVID-19 pandemic.
- Extended Challenge, in which models are tested over 2 entirely unseen cities, thus introducing a spatiotemporal domain shift in the traffic data.

Our work focuses on the Core Challenge, for which the competition organizers have provided 6 months of training data from 2019. They have tested the submissions on 100 one-hour time slots from 2020 in four diverse cities — Melbourne, Berlin, Istanbul, Chicago. For each one-hour time slot, the designed algorithms have been challenged to predict traffic conditions for the next six time

¹<https://www.iarai.ac.at/traffic4cast/> and <https://www.here.com/>

steps into the future: 5, 10, 15, 30, 45 and 60 minutes. The main difficulty of this challenge is the temporal domain shift, which appears due to the COVID-19 pandemic in 2020 that had inevitably reduced the traffic flow volume due to mobility limitations in order to abate the spread of the virus. For participants, this has imposed a problem where the train and test distributions differ significantly.

Our contributions are as follows:

- Inspired by the success of U-Net [3] in the previous competitions [4–8], we build and train four independent U-Net models for each of the core cities, which serve as a *simple, yet strong* baseline that turned out to be hard to beat.
- In attempt to further improve the score, we have conducted experiments employing DenseNet [9] and EfficientNet [10], convolutional networks pre-trained on ImageNet [11], as part of the encoder in U-Net [3].
- We further have tried a set of domain adaptation techniques, such as pseudo-labeling [12] and heuristic map pre- and post-processing to fight the temporal domain shift between the training and test data.
- Finally, we have used an ensembling method to combine the networks’ predictions, utilizing the proposed domain adaptation technique, to create the best scoring submission and have ranked **third** in the competition.

2 Data Preprocessing

The data had been collected from 4 cities: Berlin, Istanbul, Melbourne, Chicago. The competition organizers represent each city as a grid map of 495×436 pixels, where each cell aggregates the GPS measurements in a region of approximately 100×100 meters size and 5-minute time bins. Then, the dynamic traffic state at each 5-minute time bin is encoded into a grid map with 8 channels that represent traffic volume and speed in four directions (NW, NE, SW, SE). The "pixel values" of volume and speed are scaled to values from 0 to 255 with a min-max scaler and then rounded to their integer values.

Resulting input and output shapes are $T_{in} \times C \times H \times W$ and $T_{out} \times C \times H \times W$, where $T_{in} = 12$, $T_{out} = 6$, $C = 8$, $H = 495$, $W = 436$, although H and W might differ with different models, in case we pad the input image to match the working resolution of a specific model. The input represents the dynamic traffic state map over 60 minutes window, and the output includes timestamps 5, 10, 15, 30, 45 and 60 minutes after. We further reshape the input and output into three-dimensional tensors to efficiently use image-to-image models, so the resulting shapes are $(T_{in} \times C) \times H \times W$ for input and $(T_{out} \times C) \times H \times W$ for output, respectively.

As we only have 6 months of labeled data from 2019, we further randomly split the data into 4 folds, fixing one of the fold as a validation set.

Static roadmap generation The predicted traffic maps are sparse, meaning that there are pixels that never encode any traffic since there is simply no road in that region. While we expect that a large part of the road network knowledge will be inferred into the trained networks, we find that applying post-processing by simply multiplying a two-dimensional binary mask of a road graph with the network predictions yield better results. For this 2D mask, we experiment with the low-resolution static maps, provided by the competition organizers, alongside with the masks that were generated from training data the following way: we set a pixel value to 1 if the traffic volume or speed value has at least once been more than zero among the training samples, and 0 otherwise. We find that adding pixel values from 2020 test data further boosts the performance.

3 Methods

3.1 U-Net

As a strong baseline solution, we utilize a vanilla U-Net network [3], with 5 downsampling blocks and batch normalization [13] following each convolutional block with ReLU activation [14]. The U-Net network takes an image with $T_{in} \times C = 12 \times 8 = 96$ channels and outputs an image with

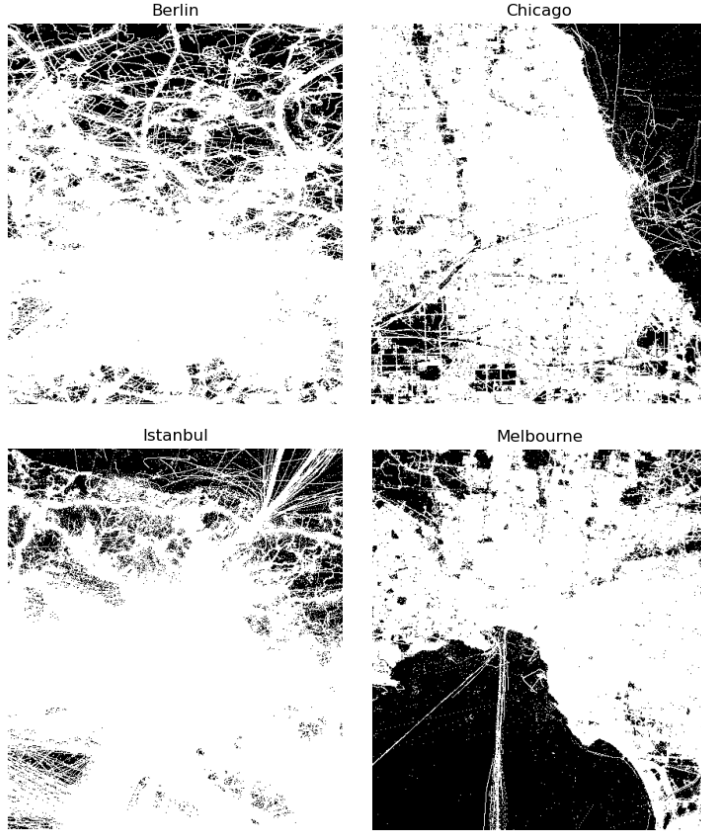


Figure 1: Generated masks of the road map from train and test data, for Berlin, Chicago, Istanbul, and Melbourne.

$T_{out} \times C = 6 \times 8 = 48$ channels. We pad the input maps to a spatial resolution of 496×448 pixels to better handle the downsampling.

Baseline training details We train an independent model for each city, with the *mean squared error* (MSE) chosen as the loss function to match the evaluation metric. PyTorch framework [15] is used for all of our experiments. We employ Adam [16] as the optimizer with the learning rate of $3e^{-4}$ and the batch size of 8. We also employ a warm-up technique by linearly scaling the learning rate from 0 to $3e^{-4}$ for the first 1k optimization steps. All our training experiments are conducted with the Mixed Precision [17] in order to decrease training time.

Results The U-Net performance on the validation set of the traffic data from 2019 per city is reported in Table 1. We furthermore evaluate on the organizer’s test set and report the scores from the leaderboard for the post-processing masks discussed in Section 2. Table 2 demonstrates that our proposed roadmap generation method yields the best result on the competition leaderboard. This post-processing technique is later used by default for all submissions.

It is worth noting that these 4 standalone U-Net models with the roadmap post-processing have been sufficient to rank third in this year’s Core Challenge, meaning that carefully tuned U-Net models provide a simple, yet strong solution for the traffic forecasting task that is robust to the temporal domain shift.

3.2 Pre-trained Encoders

Inspired by the success of employing convolutional networks, pre-trained on ImageNet [11], as encoders into U-Net [3] for a task of semantic segmentation [18], we try two different convolutional networks: DenseNet201 [9] and EfficientNet-B5 [10].

Table 1: Models’ performance over the fixed validation set from 2019 data. Mean Squared Error, the lower the better.

Model	#Params	Mean Squared Error			
		Melbourne	Berlin	Istanbul	Chicago
U-Net	31M	26.759	80.903	56.623	42.663
EfficientNet-B5 U-Net	30M	26.013	80.351	55.792	41.643
DenseNet201 U-Net	31M	25.739	78.430	-	41.158

Table 2: U-Net performance over the test set from 2020 data. Mean Squared Error, the lower the better.

Model	MSE ↓
U-Net + static city mask from competition organizers	49.73289
U-Net	49.69502
U-Net + generated static city mask from 2019 train data	49.69491
U-Net + generated static city mask from 2019 train data and 2020 test data	49.69488

EfficientNet EfficientNet [10] has been widely used in many vision tasks, such as detection [19], segmentation [20], and others, and has proved to be powerful. Thus, we employ EfficientNet as the encoder part, and keep the decoder from the original U-Net architecture, changing the number of input channels in the decoder side to match the feature maps produced by the EfficientNet. We employ the EfficientNet-B5, pre-trained on ImageNet, so the resulting network achieves almost the same number of parameters as the baseline U-Net, as shown in Table 1. The depth of the U-Net is increased by 1 to match the number of feature maps obtained from the EfficientNet backbone.

DenseNet Following the insight from the last two competitions’ winner [6, 4], we utilize the densely connected convolutional layers by employing DenseNet201 [9] pre-trained on ImageNet. Same as in EfficientNet, the decoder part is borrowed from the U-Net; however, since we omit the last dense block of DenseNet to keep the consistency across the size of the models, the resulting network’s depth is not increased compared to the baseline U-Net.

Additional training details The EfficientNet- and DenseNet U-Nets are trained with the same paramaters as the baseline U-Net, with the two differences: we increase the batch size from 8 to 16 to stabilize the training process, and also increase the padding for the input images, so the traffic maps have a spatial resolution of 512×448 pixels.

Results The performance of the two networks over the 2019 validation set can be seen in Table 1. The EfficientNet-B5 U-Net achieves a smaller MSE over all the core cities, when compared to the baseline U-Net, and the DenseNet-201 U-Net performs even better than the EfficientNet-B5 U-Net. Unfortunately, the training process of DenseNet201 U-Net on Istanbul suffered numerical instabilities, so we don’t use the DenseNet201 U-Net for this city thereafter.

However, these U-Net models with the pre-trained encoders, although showing better results during the validation on 2019 data, were not able to beat the baseline U-Net on the competition leaderboard

Table 3: Comparison of the baseline U-Net and pre-trained encoder U-Nets over the test set from 2020 data. Mean Squared Error, the lower the better.

Model	MSE ↓
EfficientNet-B5 U-Net + generated static city mask from 2019 and 2020 data	50.57098
DenseNet201 U-Net + generated static city mask from 2019 and 2020 data	49.80787
U-Net + generated static city mask from 2019 and 2020 data	49.69488

(Table 3).². This observation guided our further research direction into finding ways of bridging the gap between the 2019 training and 2020 testing data, which is further described in Section 3.3.

3.3 Domain Adaptation

Domain Adaptation is a common technique for tackling the problem when the model was trained on one domain, and tested on a different, but close target domain. As in the Core Challenge, the 2020 test data differs significantly due to a temporal domain shift, we study two Domain Adaptation strategies applicable to the 2021 Traffic4Cast Core Challenge.

Pseudo-labeling One of the studied domain adaptation approach is pseudo-labeling [12]. The idea is the following: for the unlabeled, out-of-domain data, a model predicts new labels for these samples, thus generating a new labeled data set which is later used for training the model. This procedure might be applied for several rounds. When tried for the baseline U-Net and the 2020 test data, however, our models seem to have degraded their performance on the competition leaderboard. We explain that behavior since for the 2020 test data, the temporal domain shift is also present in the model outputs.

Heuristically bridging the 2019 and 2020 data Instead of modifying the model itself, we focus on finding a way to make the 2020 testing data closer to the 2019 data, which the above-mentioned models have been trained on. We try to find a deterministic inverse transformation that would make the 2020 traffic map closer to the 2019 map, use the transformed maps to generate the model predictions, and apply the inverse transformation on predictions to return to the 2020 test data domain.

Let us denote the traffic map at a certain time step t as $M_t \in \mathbb{R}^{H \times W \times C}$, where $H = 495$, $W = 436$, $C = 8$. Then, for all the time steps from 2019 $t \in T_{2019}$ and the time steps from 2020 $t \in T_{2020}$ we calculate the mean traffic map per year the following way:

$$M_{2019} = \frac{1}{\#T_{2019}} \sum_{t \in T_{2019}} M_t$$

$$M_{2020} = \frac{1}{\#T_{2020}} \sum_{t \in T_{2020}} M_t$$

where the summation symbol denotes element-wise sum. With the mean traffic maps M_{2019} and M_{2020} for 2019 training and 2020 test data, we then calculate per-pixel and per-channel relation λ of the mean traffic map M_{2019} to the mean traffic map M_{2020} :

$$\lambda = M_{2019} \oslash M_{2020}$$

where \oslash denotes element-wise division. In case the element of M_{2020} is 0, we set the corresponding element in λ to 1. Furthermore, based on the fact that the traffic volume could not have increased in 2020, we change the outlier elements in λ which are less than 1 to 1.

We then multiply the input traffic maps from 2020 test data by λ , so the traffic maps' distribution becomes closer to the 2019 training data, run the model inference, and then multiply the predicted traffic maps by $\frac{1}{\lambda}$ ³, to bring the distribution back to the 2020 test data.

Using this *Temporal Domain Adaptation* later would result in our **best** submission on the competition leaderboard, as described in Section 3.4.

3.4 Ensembling

Having had multiple models and domain adaptation strategies by the end of the competition, we have ultimately experimented with ensembling different models and strategies together. As discussed in Section 2, for all the predictions we have utilized our proposed generated static city mask as the

²As there is no DenseNet U-Net model trained for Istanbul, we use predictions from the Istanbul baseline U-Net model

³This notation denotes that for each element x in λ , the resulting corresponding element would be $1/x$. In case $x = 0$, the value is set to 1

Table 4: Ensembled models performance over the test set from 2020 data. Mean Squared Error, the lower the better.

Ensemble	MSE ↓
Baseline U-Net, EfficientNet-B5 U-Net, DenseNet201 U-Net with TDA	49.87972
Baseline U-Net, EfficientNet-B5 U-Net, DenseNet201 U-Net	49.45481
The two of the above combined	49.37906

post-processing function. We experiment with two aggregating functions, namely mean and median ensembling, and choose the mean ensembling as it has performed slightly better.

Our three final ensembling submissions are as following:

- The mean ensemble of the predictions from the Baseline U-Net, EfficientNet-B5 U-Net, and DenseNet201 U-Net.
- The mean ensemble of the predictions from the Baseline U-Net, EfficientNet-B5 U-Net, and DenseNet201 U-Net with the *Temporal Domain Adaptation* (TDA) from Section 3.3
- The two above submission combined

We report the obtained errors on the competition leaderboard in Table 4. The ensemble of the Baseline U-Net, EfficientNet-B5 U-Net, and DenseNet201 U-Net models outperforms the standalone Baseline U-Net models; what’s more, when combining this ensemble with the predictions generated from the same models, but using the Temporal Domain Adaptation strategy, the error is further decreased to **49.37906**, which makes it our **best** submission.

4 Conclusion

Our Traffic4Cast 2021 Core Challenge solution yet again shows that U-Net [3] is a strong baseline for the traffic prediction task. We have studied further into employing convolutional encoders, pre-trained on ImageNet [11], to improve the baseline U-Net model; however, the obtained models were less robust to the temporal domain shift. In order to fight the latter, we have applied an ensembling method with the introduced *Temporal Domain Adaptation* that aims to bridge the gap between the 2019 and 2020 data. The method has ranked third on the final leaderboard.

References

- [1] David P Kreil, Michael K Kopp, David Jonietz, Moritz Neun, Aleksandra Gruca, Pedro Herruzo, Henry Martin, Ali Soleymani, and Sepp Hochreiter. The surprising efficiency of framing geo-spatial time series forecasting as a video prediction task—insights from the iarai traffic4cast competition at neurips 2019. In *NeurIPS 2019 Competition and Demonstration Track*, pages 232–241. PMLR, 2020.
- [2] Michael Kopp, David Kreil, Moritz Neun, David Jonietz, Henry Martin, Pedro Herruzo, Aleksandra Gruca, Ali Soleymani, Fanyou Wu, Yang Liu, et al. Traffic4cast at neurips 2020? yet more on the unreasonable effectiveness of gridded geo-spatial processes. In *NeurIPS 2020 Competition and Demonstration Track*, pages 325–343. PMLR, 2021.
- [3] 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.
- [4] Sungbin Choi. Utilizing unet for the future traffic map prediction task traffic4cast challenge 2020, 2020.
- [5] Jingwei Xu, Jianjin Zhang, Zhiyu Yao, and Yunbo Wang. Towards good practices of u-net for traffic forecasting, 2020.
- [6] Sungbin Choi. Traffic map prediction using unet based deep convolutional neural network, 2019.

- [7] Henry Martin, Ye Hong, Dominik Bucher, Christian Rupprecht, and René Buffat. Traffic4cast-traffic map movie forecasting–team mie-lab. *arXiv preprint arXiv:1910.13824*, 2019.
- [8] Yang Liu, Fanyou Wu, Baosheng Yu, Zhiyuan Liu, and Jieping Ye. Building effective large-scale traffic state prediction system: Traffic4cast challenge solution, 2019.
- [9] 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 4700–4708, 2017.
- [10] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [12] Amit Kumar Jaiswal, Ivan Panshin, Dimitrij Shulkin, Nagender Aneja, and Samuel Abramov. Semi-supervised learning for cancer detection of lymph node metastases, 2019.
- [13] 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.
- [14] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.
- [15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* 32, pages 8024–8035. Curran Associates, Inc., 2019.
- [16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [17] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training, 2018.
- [18] Vladimir Iglovikov and Alexey Shvets. Ternaunet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation, 2018.
- [19] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.
- [20] L. D. Huynh and Nicolas Boutry. A u-net++ with pre-trained efficientnet backbone for segmentation of diseases and artifacts in endoscopy images and videos. In *EndoCV@ISBI*, 2020.