Ivanova Ekaterina, Kolomeitsev Iurii, Kuzina Anna, Samsonov Sergei,
Shumovskaia Valentina

Predicting Semantic Segmentation of Videos

Deep Learning Project

Skolkovo Institute of Science and Technology, 2018

Contents

1 Introduction

In our project we focused on predicting future segmentations problem for different movable objects on the pictures. Several different approaches to this problem were investigated: predicting future segmentation based on the previous segmentations of the video, predicting frames based on the original frames and corresponding segmentations, adversarial networks to produce segmentation which "looks almost like" a reasonable consequence to the given. Beyond our models stands multi-scale architecture, which we wanted to evaluate on images with various objects shapes. Mostly DAVIS dataset was considered, also just to show that we are not bounded with this particular dataset, Dashcam with more homogeneous objects was used. The diversity of the objects in the first dataset was wide, that's why to obtain reasonable results and to minimize computational time, we reduced problem to the binary classification one: objects versus background.

2 Related Works

Our project was mainly based on these three papers: Natalia Neverova et al. *"Predicting Deeper into the Future of Semantic Segmentation"* [1], Michael Mathieu et al *"Deep Multi-scale video prediction beyond mean square error"* [2] and Fisher Yu, Vladlen Koltun *"Multi-Scale Context Aggregation by Dilated Convolutions"* [3].

Natalia Neverova et al. in their work try to predict segmentation maps of not yet observed video frames that lie up to a second or further in the future. They used an autoregressive CNN that learns to iteratively generate multiple frames. Authors tested their model on the Cityscapes dataset and showed that directly predicting future segmentations is substantially better than predicting and then segmenting future RGB frames.

The second work used quite similar approach, authors trained a convolutional network to generate future frames given an input video sequence and to improve the predictions in this work were proposed three different and complementary feature learning strategies: a multi-scale architecture, an adversarial training method, and an image gradient difference loss function. They compared their predictions to different published results based on recurrent neural networks on the UCF101 dataset.

The third paper refers to the dilated convolutions. The proposed CNN module uses dilated convolutions to systematically aggregate multi-scale contextual information without losing resolution. The architecture is based on the fact that dilated convolutions support exponential expansion of the receptive field without loss of resolution or coverage. Authors showed that the presented context module increases the accuracy of state-of-the-art semantic segmentation systems.

3 Problem formulation

1. Predict semantic segmentations of the future frames, including predictions for arbitrary number of steps into the future without observing true future frames themselves. In the context of this problem we will consider different formulations of S2S model: given m segmentations $S_n, S_{n-1}, \ldots, S_{n-m+1}$, predict S_{n+1} (may be proceeded recurrently further into the future). This model does not observe frames at all, only segmentations.

   Also we may consider XS2S model: given m video frames $X_n, X_{n-1}, \ldots, X_{n-m+1}$ and m segmentations $S_n, S_{n-1}, \ldots, S_{n-m+1}$ produce segmentation S_{n+1}. So it fairly predicts segmentation for 1 step into the future.

2. Improve static segmentation (just of current video frame) by capturing the recurrent structure of the video frames. For this problem we introduce XS2S++ model. In XS2S++ model: given $m+1$ video frames $X_{n+1}, X_n, \ldots, X_{n-m+1}$ and m segmentations $S_n, S_{n-1}, \ldots, S_{n-m+1}$ produce segmentation S_{n+1}. Model may be applied recurrently, if we observe an unsegmented part of the video, preceded by frames with known ground-truth segmentation.

3. Using several true segmentations of first video frames make segmentations for the whole video. This item is related with previous one: it looks possible that well-trained XS2S++ model may learn from first given number of segmentations (much less then size of the whole video frame) and provide reasonable segmentations for further frames.

4 Predicting segmentation model

4.1 Baselines

4.1.1 Last input (COPY)

When considering semantic segmentation predictions, quite a standard baseline which is sometimes difficult to outperform is just copying the last input, i.e. the last given segmentation. We will refer to this baseline as COPY in the Results section.

4.1.2 Optical Flow (OF)

A popular and yet simple way to make more clever baseline is to use optical flow estimation (refer to as OF further) - this baseline estimates optical flow between the last two inputs, and then warps the last input using the obtained flow. For its estimation a library https://github.com/pathak22/pyflow was used, which is based on Coarse2Fine method.

4.2 Architecture

Both S2S and XS2S models are multi-scale networks of [2], with 2 scaling modules (using e.g. 4 may get some gain in quality, yet will lead to substantially greater training time). Each scaling module is a four-layer convolutional network with ReLU activations, followed by Sigmoid unit to transform our input to segment $[0, 1]$. All convolutions are done with padding to keep dimensions same. If we denote by Net_1 a module of half resolution, and Net_2 - on full resolution, in the following table there will be all information about used filters and number of channels:

	NET_1	NET_2
Channels in inner layers	$32M$,$64M$,$32M$	$32M$,$64M$,$32M$
Filter sizes	$3, 3, 3, 3$	$5, 3, 3, 5$

Table 1: Architecture

Above we denoted ModSize as M.

4.3 Loss functions

We considered three different loss functions for the non-adversarial case:

- Binary Cross Entropy: $\mathcal{L}_{BCE}(\hat{S}, S) = -\left(\sum_{i,j=1} s_{i,j} \log(1 - s_{\hat{i},j}) + (1 - s_{i,j}) \log(1 - s_{\hat{i},j})\right)$
- Combined loss: sum of $l1$ and differences losses: $\mathcal{L}_c(\hat{S}, S) = \mathcal{L}_{l1}(\hat{S}, S) + \mathcal{L}_{gdl}(\hat{S}, S)$, where

$$\mathcal{L}_{l1}(\hat{S}, S) = \sum_{i,j=1} |s_{i,j} - s_{\hat{i},j}|$$

$$\mathcal{L}_{gdl}(\hat{S}, S) = \sum_{i,j=1} ||s_{i,j} - s_{i-1,j}| - |s_{\hat{i},j} - s_{\hat{i-1},j}|| + ||s_{i,j} - s_{i,j-1}| - |s_{\hat{i},j} - s_{i,\hat{j-1}}|||$$

- Dice loss, i.e. a continuous analogue of the Dice index (Jaccard index): $\mathcal{L}(\hat{S}, S) = -\log d(\hat{S}, S)$, where

$$d(A, B) = \frac{2\sum_{i,j=1}^{n} a_{i,j} b_{i,j}}{\sum_{i,j=1}^{n} a_{i,j}^2 + \sum_{i,j=1}^{n} b_{i,j}^2}$$

4.4 Adversarial Networks

For adversarial training we used architecture from [2]: the discriminator D takes a sequence o semantic segmentations and is trained to predict probabilities that the last segmentation in the sequence are generated by generator G. At the same time generator is trained to produce segmentations, coherent with the last observed input segmentation, and, hence, to trick the discriminator.

- Training D: architecture of D and G is also multi-scaling, with 2 convolutional networks as scaling units. If D_1 is a unit on half resolution, D_2 - on full resolution, then to outputs of both D_1 and D_2 we add fully connected layers in order to produce finally 1 number for D_1 and one for D_2 - probability for last segmentation to be generated or not. So, in fact, we train each layer of the net to discriminate if their resolution and then

$$\mathcal{L}_{adv}^{D}(S, \hat{S}) = \sum_{k=1}^{N_{\text{scales}}} \mathcal{L}_{bce}(D_k(S, S_{t+1}), 1) + \mathcal{L}_{bce}(D_k(S, G(S), 0))$$

  Where we denoted (S, S_{t+1}) - sequences of segmentations followed by true segmentation and $(S, G(S)$ - sequence of segmentations, followed by generated one. Note that in each minibatch of this type only the last picture is artificial. Architecture is drawn below:

	D_1	D_2
Channels in inner layers	$32M$	$32M$,$32M$
Filter sizes	$3, 3$	$3, 3, 3$
Fully connected	$(N, 256), ReLU, (256, 1)$	$(N, 256), ReLU, (256, 1)$

Table 2: Discriminator

- Training G: Let (S, S_{t+1}) be a different (from previous one) sample from the dataset, then, keeping weights of D fixed, we compute

$$\mathcal{L}_{adv}^{G}(S, S_{t+1}) = \sum_{k=1}^{N_{\text{scales}}} \mathcal{L}_{bce}(D_k(S, G(S), 1)$$

  Yet this is not the final requirement on the generator: we also need it to produce segmentation, close to original, so we add regularization, and perform optimization step to minimize

$$\mathcal{L}_{\text{fin}} = \lambda_{\text{adv}} \mathcal{L}_{adv}^{G}(S, S_{t+1}) + \lambda_{\text{predict}} \mathcal{L}_{l1}(G(S), S_{t+1}) \tag{1}$$

  or

$$\mathcal{L}_{\text{fin}} = \lambda_{\text{adv}} \mathcal{L}_{adv}^{G}(S, S_{t+1}) + \lambda_{\text{predict}} \left(\mathcal{L}_{l1}(G(S), S_{t+1}) + \mathcal{L}_{gdl}(G(S), S_{t+1})\right) \tag{2}$$

  In our evaluations we will refer to these models as S2S Adv-1 for loss (1) and S2S Adv-2 for loss (**??**) respectively. Architecture of G is in table below:

	G_1	G_2
Channels in inner layers	$32M$,$64M$,$32M$	$32M$,$64M$,$32M$
Filter sizes	$3, 3, 3, 3$	$5, 3, 3, 5$

Table 3: Generator

Above we denoted ModSize as M.

4.5 Predicting deeper into the future

In the original paper [1] two main extensions were considered to predict deeper into the future:

- Batch approach: the output of the network is just scaled to predict m segmentations instead of 1. This approach ignores the recurrent structure of the data (for instance, similarity in dependencies between $(S_{1:n}, S_{n+1})$ and $(S_{2:n+1}, S_{n+2})$. Hence, this one was not considered in our further studies.
- Autoregressive approach: iteratively apply a model that predicts a segmentation for a single step into the future using its prediction for frame $n+1$ as an input to predict the segmentation of the $n+2$ frame, and so on.

5 Experiments

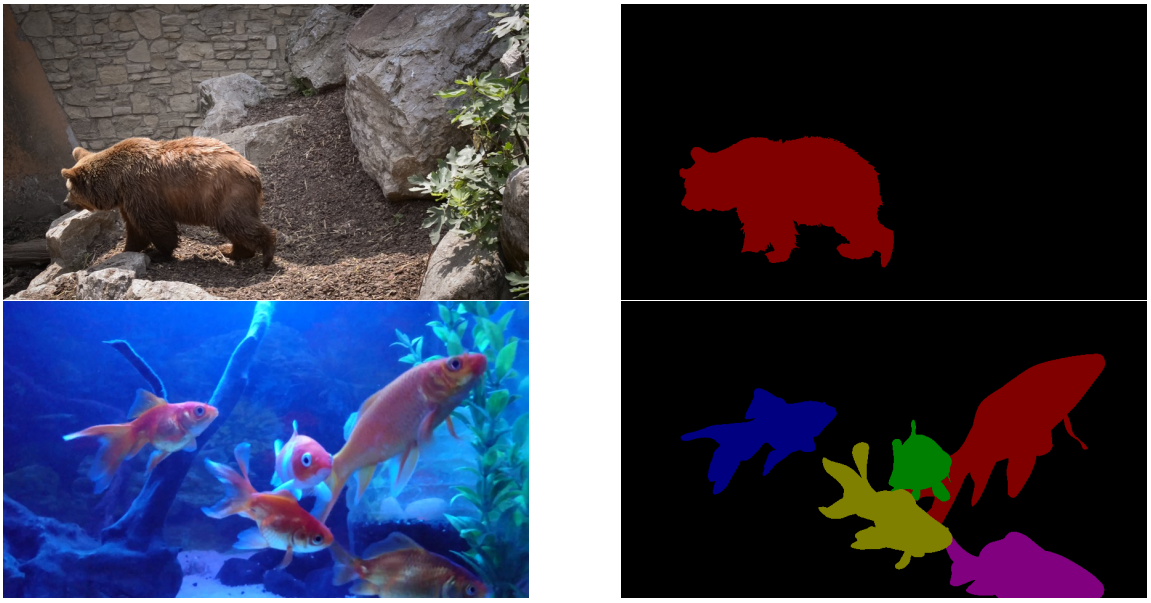


Figure 1: Some examples from Davis datasets

5.1 Davis dataset

DAVIS (Densely-Annotated Video Segmentation) dataset (is a part of DAVIS Challenge on Video Object Segmentation). On each video there is one movable object (bear, plane, surfer, etc.), e.g. figure 1, the semantic segmentation of which we want to predict. Dataset consists of sequences of frames and true segmentations. Resolution of images is 854×480. All in all training part of the dataset are 60 videos with totally 4200 frames, validation part - 30 videos with 2000 frames.

5.2 Dashcam dataset

We also applied model to another dataset for comparison: data from kaggle competition CVPR 2018 WAD Video Segmentation Challenge. It is also a set of videos, but contrary to DAVIS, they are all of the same nature, depicting videos from a dashcam (see figure 2 for example). Here, we also tried to predict segmentation of movable objects on the road (such as cars, bicycles or, pedestrians)etc). Yet for the comparability with the previous dataset, we again reduced the problem to binary segmentation case: movable object versus background.

The whole dataset was of size 90 gigabytes approximately, so we took video from only one camera - it gave us 4084 frames with known segmentation, and we splitted it into train and validation parts

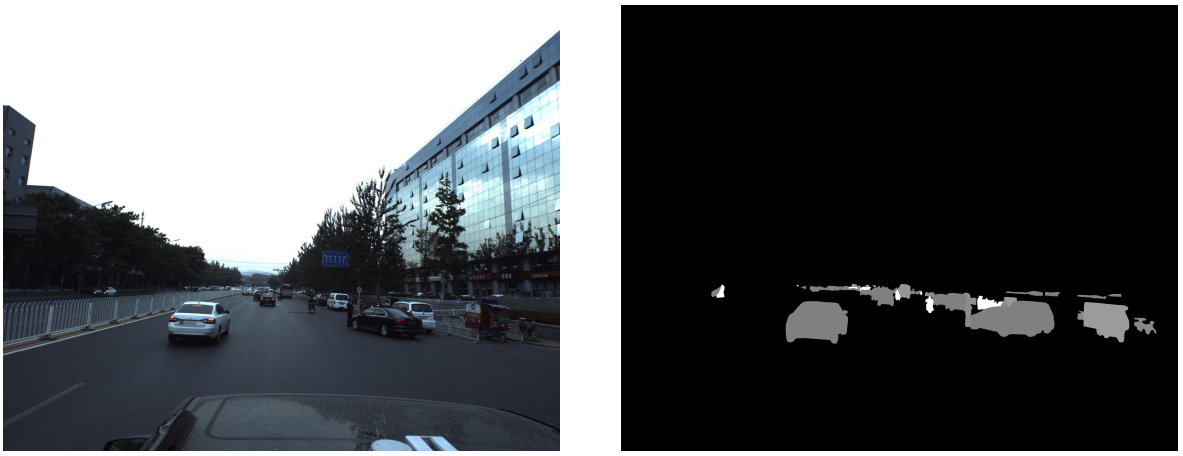


Figure 2: Example from dashcam dataset

by dividing it to subsequences of length 100, moving first 70 frames to train, and following 30 frames to validation.

Originally resolution of each image was extremely high — 2710×3384, much higher then for DAVIS dataset, hence we simply downsampled both Test and Validation subsets with MaxPool and kernel of size $(8, 8)$, getting 339×420 segmentations, and our experiments were conducted in this resolution.

5.3 Metrics

As a measure of obtained segmentation quality we will use intersection-over-union metrics:

$$IoU(S, \hat{S}) = \frac{S \cap \hat{S}}{S \cup \hat{S}}$$

As we are going to solve binary classification problem, here we are interested in IoU[object] and IoU[background], for sake of notations - $IoU[0]$ and $IoU[1]$. In addition, we report accuracy scores.

5.4 Technical remarks

1. Most of the experiments without GANS were conducted with Adam optimizer and default learning rate 1^{-4}.
2. For Adversarial learning also Adam was used, with learning rate 1^{-2} for discriminator and $2 \cdot 10^{-3}$ for generator, decreasing each epoch by factor 0.95.
3. If contrary is not specified, $S2S$ model train on sequences of length 4 to predict the next seg (5-th), XS2S and XS2S++ takes as input 2 frames, 2 segmentations and 3 frames, 2 segmentations respectively.
4. Different models were trained with different batch sizes, 16 for the smallest, 2 for the deepest. Yet batch size does not affect results significantly.
5. If contrary is not specified, default $ModSize = 1$ was used.
6. All reported scores are obtained on the validation dataset, iterating through it all and then averaging.
7. For model S2S Adv-1: $\lambda_{\text{adv}} = 0.05$ and $\lambda_{\text{predict}} = 1$
8. For model S2S Adv-2: $\lambda_{\text{adv}} = 0.1$ and $\lambda_{\text{predict}} = 1$
9. Each model was trained separately on Tesla K80 GPU.

6 Results

6.1 Loss functions comparison

Here the goal is to find out, whether different loss types significantly affect quality results. So 3 models, same except loss function were considered: S2S model, 1-step-ahead predictions, 50 training epochs, each longs about 15 minutes. Results are in table 13.

	COPY	BCE	GDL+l1	Dice
Accuracy	0.9782	0.9836	0.9833	0.9836
mIoU	0.8792	0.9072	0.9055	0.9076
IoU[0]	0.9737	0.9801	0.9797	0.9801
IoU[1]	0.7846	0.8343	0.8314	0.8351

Table 4: losses

From the results above if follows that, at least for DAVIS dataset, the actual difference between losses is not very crucial - since that, we will use Binary Cross-Entropy as our default choice in further evaluations.

6.2 S2S 1-step-ahead predictions

All models evaluated in this subsection are with $ModSize = 1$, and we stand S2S-dil model for the one with dilated convolutions. Each of the GANs took about 40 minutes for 1 epoch, so we trained them for 25 epochs each, other nets - for 50 epochs. Results are in table 5, graphics with learning information in 3.

	COPY	OF	BCE	GDL+l1	Dice	Adv-1	Adv-2	S2S-dil
Accuracy	0.9782	0.9642	0.9836	0.9833	0.9836	0.9842	0.9840	0.9851
mIoU	0.8792	0.8232	0.9072	0.9055	0.9076	0.9096	0.9087	0.9129
IoU[0]	0.9737	0.9575	0.9801	0.9797	0.9801	0.9808	0.9806	0.9818
IoU[1]	0.7846	0.6889	0.8343	0.8314	0.8351	0.8385	0.8368	0.8440

Table 5: S2S 1-step-ahead

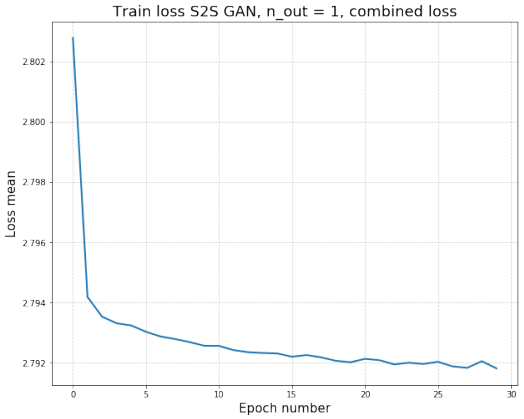
Note that best results are obtained for S2S-dil (with dilated convolutions), which even outperforms GAN models. It is probably explained by the fact that kernel size in convolutional layer somehow bound our opportunity to find long-term dependencies, and this affect is smaller for dilated convolutions. Low result for optical flow method may be explained by too diverse moving routes for different objects in different videos.

6.3 XS2S++ versus static baselines

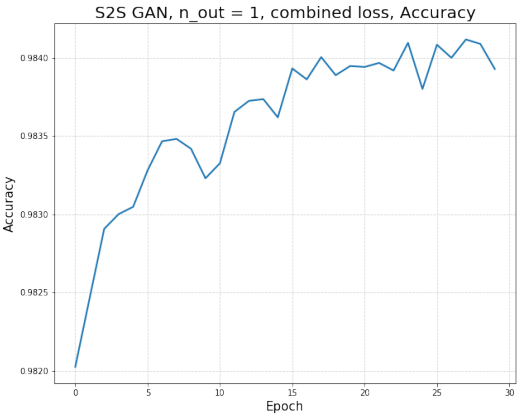
Here we consider XS2S++ model for 1 frame segmentations, compared with COPY baseline and S2S model predictions. XS2S++ model here is used with ModSize = 2. Results are in table 6.

	COPY	S2S	XS2S++
Accuracy	0.9782	0.9842	0.9885
mIoU	0.8792	0.9102	0.9334
IoU[0]	0.9737	0.9808	0.9860
IoU[1]	0.7846	0.8396	0.8808

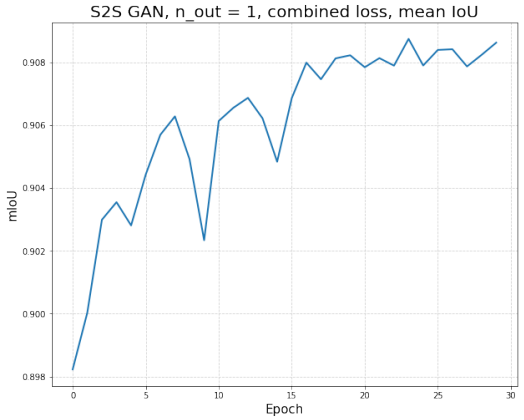
Table 6: XS2S++1step



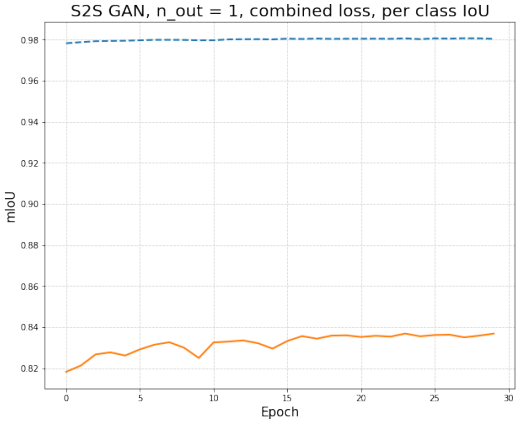
(a) train loss



(b) per-pixel accuracy



(c) miou



(d) iou

Figure 3: S2S Adv-2 GAN for 1-step-ahead

6.4 5-steps-ahead mode comparison

Although this is not absolutely fair, we will compare S2S model with XS2S and XS2S++ for 5-steps-ahead segmentation. Here we are interested to estimate, how quickly mean IoU rate is decreasing, and how S2S predicting model behaves in comparison with ones, observing real video frames. Results are in table 7.

	COPY	S2S	XS2S	XS2S++
Accuracy[1 frame]	0.9778	0.9830	0.9823	0.9861
Accuracy[3 frame]	0.9538	0.9622	0.9697	0.9724
Accuracy[5 frame]	0.9373	0.9480	0.9600	0.9615
mIoU[1 frame]	0.8786	0.9048	0.9013	0.9210
mIoU[3 frame]	0.7848	0.8183	0.8512	0.8639
mIoU[5 frame]	0.7323	0.7708	0.8184	0.8255
IoU[1][1 frame]	0.7839	0.8304	0.8240	0.8590
IoU[1][3 frame]	0.6239	0.6814	0.7384	0.7606
IoU[1][5 frame]	0.5370	0.6027	0.6838	0.6959

Table 7: AR5

6.5 S2S, autoregression 5-steps-ahead

Here we considered 3 different settings for S2S AR5 model:

- S2S AR5 with BCE loss (ModSize = 2)
- S2S-dil AR5 with BCE loss (ModSize = 2)
- S2S Adv-2 AR5 (ModSize = 1), chosen instead of Adv-1 due to more stable loss behavior.

Results are in table 8. Thus, dilated convolutions improve result not only in short-term, but also in

	Baseline	S2S AR5	S2S-dil AR5	S2S Adv-2 AR5
Accuracy[1 frame]	0.9778	0.9830	0.9844	0.9839
Accuracy[3 frame]	0.9538	0.9622	0.9644	0.9628
Accuracy[5 frame]	0.9373	0.9480	0.9495	0.9456
mIoU[1 frame]	0.8786	0.9048	0.9121	0.9085
mIoU[3 frame]	0.7848	0.8183	0.8293	0.8236
mIoU[5 frame]	0.7323	0.7708	0.7757	0.7693
IoU[1][1 frame]	0.7839	0.8304	0.8431	0.8366
IoU[1][3 frame]	0.6239	0.6814	0.7011	0.6914
IoU[1][5 frame]	0.5370	0.6027	0.6110	0.6029

Table 8: S2S AR5

mid-term perspective.

6.6 Model size influence

Consider S2S problem with BCE loss for 1-step-ahead predictions, and note that $ModSize$ is a multiplicative parameter in number of convolutions, i.e. $ModSize = 2$ corresponds to twice larger number of channels in convolution layers compared to $ModSize = 1$. Note that data below clearly indicates that simply making model bigger does not lead to significant gain in quality, but the bigger model trained twice longer then the smaller one, for same number of epochs. Results are in table 9.

	ModSize = 1	ModSize = 2
Accuracy	0.9836	0.9842
mIoU	0.9072	0.9102
IoU[0]	0.9801	0.9809
IoU[1]	0.8343	0.8396

Table 9: ModSize

6.7 Mid-term predictions with transfer learning

Here we compare S2S AR5 model trained in two different setups: 25 epochs from scratch (each epoch tooks about 1 hour on Tesla K80 GPU), and pre-trained model (S2S fine-tuned): first 25 epochs from scratch for 1-step-ahead predictions (15 minutes per epoch Tesla K80 GPU), then 4 epochs in AR5 mode to converge. Note that results are comparable, despite the first model tooks 2.5 times less time to converge. Results contain in table 10.

	S2S fine-tuned	S2S AR5 from scratch
Accuracy	0.9836	0.983
mIoU[1 frame]	0.9074	0.9048
mIoU[3 frame]	0.8234	0.8183
mIoU[5 frame]	0.7685	0.7708
IoU[1][1 frame]	0.8347	0.8304
IoU[1][3 frame]	0.6908	0.6814
IoU[1][5 frame]	0.5973	0.6027

Table 10: Transfer learning

6.8 Long-term predictions with transfer learning

Now consider problem of prediction in S2S AR10 mode, i.e. how to predict semantic segmentation for the next 10 video frames without observing them. Clearly, learning from scratch will last too long, so we compared 2 models:

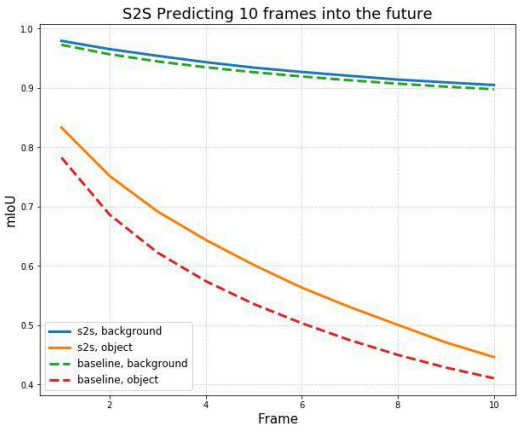
- S2S AR10 fine-tuned from S2S 1-step-ahead, $ModSize = 1$
- S2S AR10 fine-tuned from S2S AR5 (i.e. 5-steps-ahead), $ModSize = 2$

In both settings BCE loss was used both for pre-training and fine-tuning, both models were fine-tuned for $n =$ epochs.

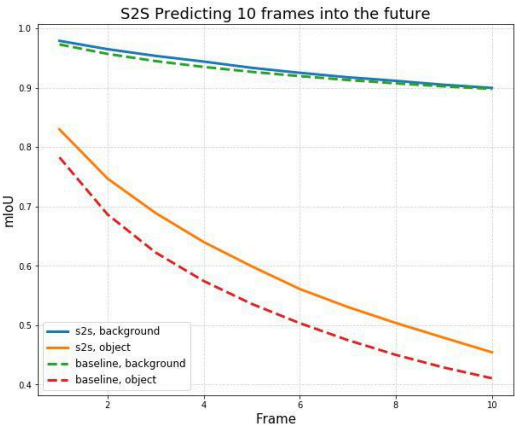
	COPY	S2S AR10 from AR1	S2S AR10 from AR5
Accuracy[1 frame]	0.9778	0.9832	0.9826
Accuracy[5 frame]	0.9373	0.9443	0.9434
Accuracy[10 frame]	0.9089	0.9165	0.9120
mIoU[1 frame]	0.8786	0.9063	0.9044
mIoU[5 frame]	0.7323	0.7680	0.7663
mIoU[10 frame]	0.6542	0.6756	0.6770
IoU[1][1 frame]	0.7839	0.8332	0.8298
IoU[1][5 frame]	0.5370	0.6017	0.5992
IoU[1][10 frame]	0.4105	0.4462	0.4543

Table 11: Transfer learning S2S AR10

Note that the results are almost the same, so for 10-steps-ahead predictions model size almost didn't affect quality.

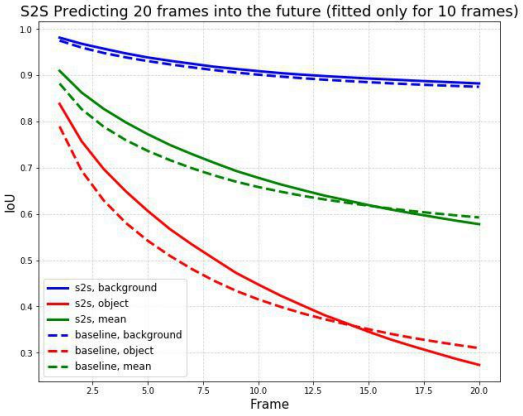


(a) miou

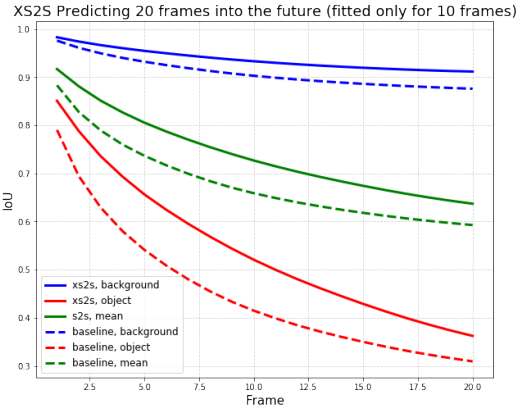


(b) per-pixel accuracy

Figure 4: 10 steps ahead for S2S predictions



(a) S2S AR20 vs COPY baseline



(b) XS2S++ AR20 vs COPY baseline

Figure 5: 20 steps ahead S2S and XS2S++

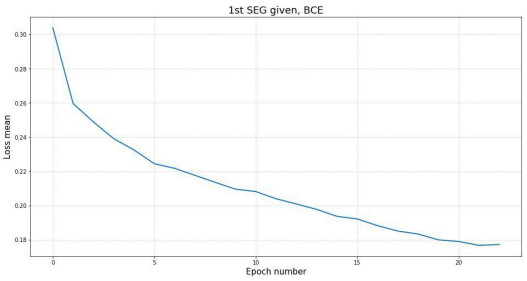
Here are also the results for *meanIoU* decay rate while number of steps increases: at the figure 4 are plots for S2S and XS2S++ models.

To find out, when we will find our baseline, let us consider the same S2S model to predict for 20 steps ahead, without fine-tune, just as XS2S++, and compare them both with COPY baseline (figure 5 for reference):

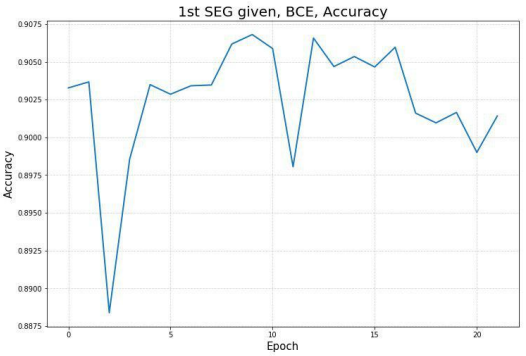
6.9 XS2S++, only first segmentations given

Since XS2S++ model is successful enough even with long-term predictions, we may try to obtain segmentations for the whole dataset using only ground-truth segmentations for first frames in the following manner:

- SEG1 - generates segmentation of frame n provided frames 1,n-1,n, and segmentations 1
- SEG2 - generates segmentation of frame n provided frames 1,2,n-1,n, and segmentations 1,2

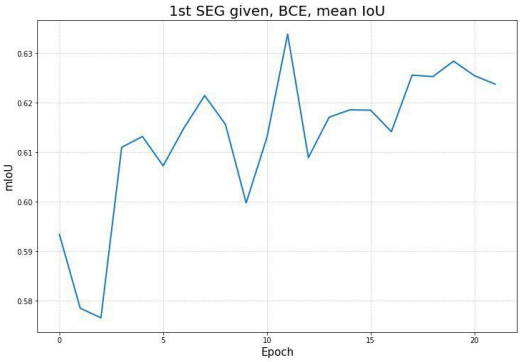


(a) train loss

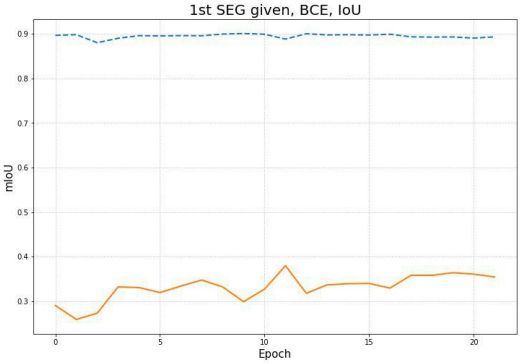


(b) per-pixel accuracy

h!



(c) miou



(d) iou

Figure 6: XS2S++ with only first 2 segmentations given

Results are attached in table 12. Note that, unfortunately, they are not promising enough - object detection is very far from perfect. For more detailed information look at 6. So capturing the way how object moves from the first segmentations is really very hard.

	SEG1	SEG2
Accuracy	0.8980	0.8974
mIoU	0.6338	0.6326
IoU[0]	0.8883	0.8876
IoU[1]	0.3794	0.3776

Table 12: FirstSeg

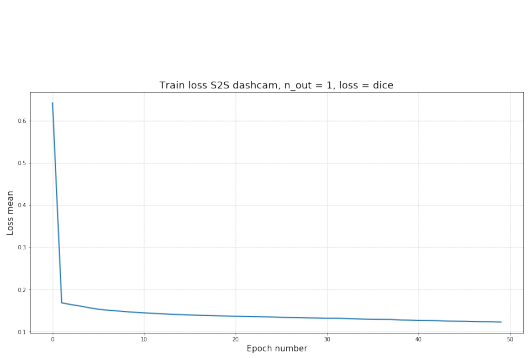
6.10 What about another dataset?

Now let us apply S2S (with Dice loss),S2S-Adv-1, S2S-dil to the subset of Dashcam dataset, and compare with the copy-last-input baseline:

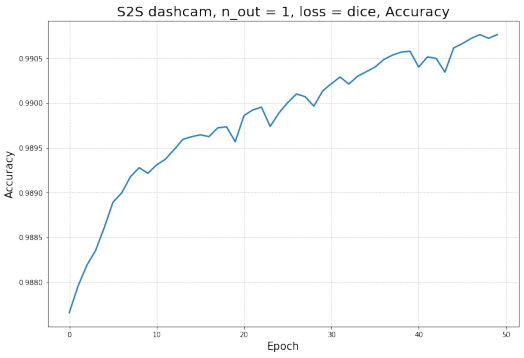
	COPY	S2S-dice	S2S-Adv-1	S2S-dil (dice)
Accuracy	0.9868	0.9907	0.9920	0.9909
mIoU	0.8238	0.8822	0.8888	0.8901
IoU[0]	0.9864	0.9903	0.9916	0.9905
IoU[1]	0.6611	0.7741	0.7859	0.7896

Table 13: Losses

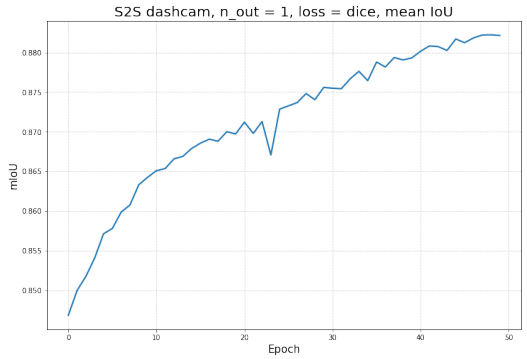
So, for this dataset it is easier to learn information about how object move - probably because they are more homogeneous in their nature.



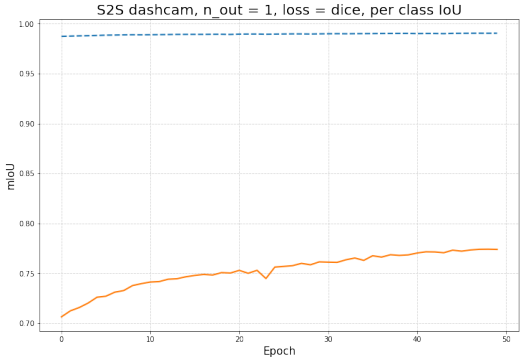
(a) train loss



(b) per-pixel accuracy



(c) miou



(d) iou

Figure 7: Dashcam, S2S, 1-step-ahead

7 Conclusion

We compared on DAVIS dataset different approaches to predict future segmentations - multi-scale architectures, fully based on convolutional units, both with and without dilated convolutions, adversarial learning in the spirit of generating likelihood segmentations. We may conclude that dilated convolutions architectures shows themselves best for short-term and mid-term predictions, even outperforming GAN models. Partially it may be explained by the lack of computational resources - GANs are known to take long time to train, and single GPU resources may be not enough. For 1-step-into-the-future, this result was confirmed by Dashcam dataset predicitons.

We estimated the speed of predictions degrading when considering more and more steps into the future, and found, that pure S2S predictions start to loose against copying baseline after 15 frames ahead.

We tuned XS2S++ model for different number of frames ahead segmentations, and results for the first frames are very good - obtained outcomes may be used to improve quality of static segmentation methods, which ignores the recurrent structure of the video and information, for example for the case of the known ground-truth segmentation for part of the dataset. Improvement may be sufficient for first frames after the ground-truth segmentation is no longer accesible, later the accumulated error will increase, and XS2S++ will start loosing to static methods.

8 Contributions

Ekaterina:
Fitting XS2S++ model which used only first segmentations, implementation of data generator for Dashcam dataset, creation of gif visualization based on predictions.

Iurii:
Implementation of S2S-dilation model, fitting S2S, XS2S models for single frame predictions on Davis, transfer learning vs. training from scratch comparison, modsize comparison.

Anna:
Implementation of XS2S and XS2S++ models, implemented data generator for Davis dataset and fitting XS2S++ model for single frame predictions.

Sergei:
Implementation of Adversarial network, loss functions comparison and implementation, fitting S2S, XS2S models in AR5, AR10 modes.,Implementing training/Validation.

Valentina:
Implementing and computing COPY and OpticalFlow baselines on 2 datasets, implementation of S2S model, fitting S2S models on Dashcam dataset.

All:
Making first and final presentations, writing report.

9 GitHub repository with code and Gifs

In our repository on GitHub: `https://github.com/kolomeytsev/predicting_semantic_segmentation_of_videos/` you can see our code. Also, in README.md we have attached several Gifs that show the performance of the models: S2S AR10 and XS2S++ AR10.

10 Third-party code used

- Optical flow lib https://github.com/pathak22/pyflow

References

[1] Natalia Neverova, Pauline Luc, Camille Couprie, Jakob J. Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. *CoRR*, abs/1703.07684, 2017.

[2] Michaël Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2015.

[3] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.