Recognizing Gates via Semantic Segmentation Based on Fully Convolutional Networks in Autonomous Drone Racing

Cheng Liu
Control & Simulation, Faculty of Aerospace Engineering
Delft University of Technology, the Netherlands
c.liu-10@tudelft.nl

***Abstract*—Recognizing gates in images from drone's camera is essential in passing through way-points in a vision-based drone racing task. In this paper, we trained fully convolutional neural networks (FCN) with variant architectures in an end-to-end style on the Artificial Intelligence Robotic Racing (AIRR) dataset for drone racing to recognize gates from pixels-to-pixels. To alleviate computation consumption, we proposed a concise encoder neural network named VGG5, which can operate at higher speeds while achieving equivalent predicting metric scores as other more complex models such as VGG11. To solve the information loss problem due to the coarse stride of single-stream decoding, we adapt skip-layer structures that combines semantic information from a deep, coarse layer with appearance information from a shallow, fine layer to produce accurate and detailed gate segmentations. As demonstrated in evaluation results, the proposed VGG5 based FCN model can perform very well on recognizing gates in detail, and skip-layer structure plays an essential role in doing this. Source code can be found in this link[1].**

***Index Terms*—fully convolutional networks, drone racing, computer vision, deep learning**

I. Introduction

Autonomous drone racing brings up fundamental challenges in robotics. In these drone racing competitions, the drone has to pass through lined-up gates as quickly as possible under severe computation constraints without any collision. Therefore, it is critical to detect gates reliably and efficiently using computer vision techniques. Due to the complications such as varying lighting conditions and gates seen overlapped, traditional image processing algorithms based on color and geometry of the gates tend to fail during the actual racing. Recently, applying *Artificial Intelligence (AI)* like *deep learning* [1] to autonomous drone racing has become an upward trend due to its reliability, data-driven characteristic and capability of generalizing to unseen environments. In this paper, we present a deep learning model for gate recognition using semantic segmentation based on *Fully Convolutional Networks (FCN)* [2]. Specifically, we use *VGG* [3] convolutional neural networks as the feature encoder, and upsample these features to the original image dimension by deconvolution networks in the decoder. Oriented to embedded systems with resource constraints, we simplified the original VGG which was designed for large-scale image recognition to a more concise encoder architecture which we call *VGG5*, achieving higher inference speed and smaller memory storage without losing perception accuracy. To reconstruct the information lost in the upsampling process as a consequence of coarse stride, we adapt the skip-layer structures as in the original FCN paper [2] to produce more accurate and detailed gate segmentations especially at object boundaries. It can be found in evaluation experiments that this skip-layer structure highly improved the prediction accuracy given the same VGG architecture. This encoder-decoder structure with skip-layers is shown in Fig. 1 and will be explained in details in Section III.

[1]https://github.com/chengliu-LR/gate-detection-FCN.git

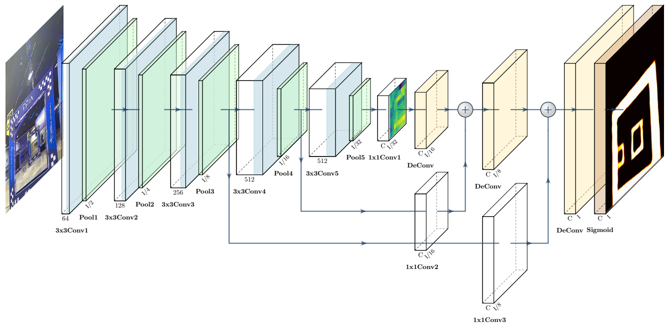

Fig. 1. FCN-8s Gate Recognition Architecture based on VGG5. Note that there are element-wise connections to utilize the detailed perception in shallow layers and larger receptive fields in deep layers.

II. Fully Convolutional Networks

A. An encoder-decoder structure

Our goal of gate recognition is to take the gate image and output a segmentation map where each pixel contains a one-hot-encoding class prediction indicating whether it is a gate or not. This is also known as *dense prediction*. A naive approach towards constructing a neural network architecture for this task is to simply stack a number of convolutional layers (with same padding to preserve dimensions) and output a final segmentation map. This directly learns a mapping from the input image to its corresponding segmentation through the successive transformation of feature mapping. However, it's quite computationally expensive to preserve the full resolution throughout the network. One popular approach for image segmentation models is to follow an encoder-decoder structure where we downsample the spatial resolution of the input by using stacked convolutional layers like VGG, developing lower-resolution feature mappings to keep filters small and computational requirements reasonable, and then upsample the feature representations into a full-resolution segmentation map to match the original input's resolution.

B. In-Network Upsampling

A natural way to upsample is using transposed convolutional neural networks [4], or deconvolutions [2]. Upsampling is performed in-network for end-to-end learning by backpropagation from pixelwise loss functions. Whereas convolutional layers reduce the input resolution or keep them unchanged, deconvolution serves for increasing input spatial resolution. In a deconvolution, we take a single value from the low-resolution feature map and multiply all of the weights in the deconvolution kernel by this value, projecting those weighted values into the output feature map. By combining multi-layers of convolution and deconvolution, we obtain fully convolutional networks (FCNs). FCNs naturally operate on an input image and produce an output of corresponding spatial dimensions.

III. Gate Recognition Architecture

A. Simplified VGG for Perception

We adapt VGG architecture as the backbone of our encoder network. The original VGG model was designed for large-scale image recognition with very deep convolution structures, thus not suitable for the relatively small size and less diversity of the gate dataset used in this paper. Considering this, we simplified the complex VGG to more concise ones. Specifically, We discarded the fully-connected layers in original VGG and connect it with 1×1 convolutions to reduce the dimension of feature maps to match the number of classes C (for this gate recognition task, $C = 1$). Moreover, we tailored one of the VGG model which is called *VGG11* in [3] by removing some convolutional layers in each block and resulted in our VGG5 model. As shown in Fig. 1, There are overall 5 convolutional layers with 3×3 kernel and stride 1 for gate perception. The padding is set to be 1 to preserve the spatial resolution after each convolutional layer. As the network goes deeper, the number of features increases from 3 (RGB channels of the original image) to 512. Each layer is equipped with a *rectified-linear unit (ReLU)* as the activation function (represented with blue color in Fig. 1) and a max-pooling layer to reduce the spatial resolution of the feature map and create an invariance to small shifts and distortions (represented in green). The spatial resolution of the input is halved after each max-pooling layer with a 2×2 kernel and stride 2. With an input spatial resolution of $I\times I$, VGG5 outputs a $\frac{I}{32}\times\frac{I}{32}$ feature map with C channels.

B. Skip-layer Connections

A direct way to reconstruct the input spatial dimension is connecting the last layer of VGG5 to a deconvolutional layer with a stride of 32, which resulted in *FCN-32s*. After training, we observed that FCN-32s can achieve a decent prediction accuracy but lacks the ability to predict detailed gate boundaries. This is due to the coarse stride of the deconvolutional layer: as encoder reduces the resolution of the input by a factor of 32, decoder struggles to produce fine-grained segmentations by just upsampling the last layer of VGG5, which is a coarse feature map.

To solve this problem, we adapted the skip-layer structure proposed in [2] to combine coarse, semantic and local, appearance information to refine prediction. Firstly, we filter the second last layer of VGG5 with a 1×1 convolutional layer to reduce the number of features to C, and upsampling the last layer of VGG5 with stride of 2 using deconvolution. After these operations, we are able to add these two layers in an element-wise manner to generate a feature map with spatial dimension of $\frac{I}{16} \times \frac{I}{16}$. Furthermore, we upsample this new feature map using another deconvolutional layer with stride 16 to finally reconstruct the input spatial resolution. This model is called *FCN-16s*. With this skip-layer connection, the decoder can make local predictions while retaining high-level semantic information, leaving the architecture of encoder unchanged. Based on FCN-16s, we also built another model called *FCN-8s* with 2 skip-layer connections. In FCN-8s, we add another skip-layer connection from the third last layer of VGG5 to a $\frac{I}{8} \times \frac{I}{8}$ feature map. This feature map is upsampled with stride 2 from the FCN-16s model output with the 16-strided deconvolutional layer omitted. The input spatial resolution is then reconstructed by applying a final deconvolutional layer with stride 8. FCN-32s, FCN-16s and FCN-8s are all equipped with a non-linear *sigmoid* final layer to restrict the value of each pixel in range $[0, 1]$ for pixelwise logistic regression. The detailed architecture of FCN-8s is demonstrated in Fig. 1.

IV. Experimental Framework

A. Data Preprocessing

We train FCNs on a front camera image dataset collected from the Artificial Intelligence Robotic Racing (AIRR) Drone Racing League [5]. Each image in the dataset is labeled with a gate segmentation mask for supervised learning. The dataset is separated into training set and test set randomly at a ratio of 9:1. Before being fed to FCN, images should be scaled to fit the input shape of the neural network, and also the masks. However, since we are re-mapping the predicted pixel categories back to the original-size input image, it would be difficult to do this precisely, especially in segmented regions with different semantics. To avoid this problem, instead of rescaling, images and masks are randomly cropped to be consistent in spatial resolution. Every time the dataset is sampled, the random crop will be applied. This can also be considered as data augmentation.

The cropped image spatial resolution is set to be 288×288 in two main reasons: i) The dataset consists of images with divergent spatial resolutions of which the minimum image height/width is 315. The cropped image size is supposed to be smaller than it to make sure random crop can be implemented correctly. ii) Since overall five 2-strided max-pooling with 2×2

kernel is used in FCNs, 288 is the maximum number which is not only smaller than 315 but also can be halved five times with no remainder.

B. Training

1) Loss Function: The most commonly used loss function for semantic segmentation is pixelwise cross entropy loss. For gate recognition with only the gate to be distinguished from the background, we choose *Binary Cross Entropy (BCE)* as the loss function for training, which is defined as

$$L_{bce}(\mathbf{y}, \hat{\mathbf{y}}) = -\mathop{\mathbb{E}}_{i \in B}\{\mathbf{y}_i \log \hat{\mathbf{y}}_i^T + (1-\mathbf{y}_i)\log(1-\hat{\mathbf{y}}_i^T)\} \quad (1)$$

where B represents all samples from one batch (the batch size is set to be 16 for training), $\mathbf{y}_i$ is one-dimensional flatten mask matrix with each element a binary label in $[0, 1]$. $\hat{\mathbf{y}}_i$ is the FCN prediction filtered with sigmoid non-linear function.

2) Variant FCN Architectures: In the training process, we trained variant FCNs with different architectures. Concretely, we first trained FCN-32s, FCN-16s and FCN-8s built on VGG5. After that, in case of an underfitting of the relatively simple VGG5 model for this task, we replaced VGG5 with VGG11 to see if a more complex perception model can further decrease BCE loss. To potentially achieve a faster and more stable learning process, we added *batch normalization (BN)* [6] after each convolutional layer in VGG11 model. However, in our evaluation experiments, adding BN layers brings nearly no benefits on evaluation performance but only a smaller initial and final training loss. Besides, BN can also increase computational consumption. As a consequence, we didn't add it to VGG5. All FCN parameters are randomly initialized except deconvolutional layers, which are manually initialized by bilinear interpolation with the parameters allowed to be learned. Bilinear interpolation computes each output y_{ij} from the nearest four input pixels by a linear map.

3) Optimization: We firstly resorted to *Stochastic Gradient Descent (SGD)* with momentum to train FCNs but it required quite a lot effort of hyperparameter tuning and always got stuck in local optimums. So instead, we used Adam [7] optimizer with learning rate $\alpha = 10^{-4}$, exponential decay rates $\beta_1 = 0.9, \beta_2 = 0.999$ and numerical stabilizer $\epsilon = 10^{-8}$. We did independent training experiments for each FCN architecture. Every training experiment is executed for at least 1000 epochs to make sure the loss function has been converged. In each training epoch, we iterate through the training dataset with shuffling and do batch-wise optimization in parallel.

C. Evaluation

1) Segmentation Metrics: We choose mean *intersection over union (IoU)* and *pixel prediction accuracy* over test set as our metrics to evaluate prediction accuracy for each trained FCN. `True/False` threshold of 0.5 is used. Let n_{gg} as the number of true positive prediction of gate pixels, and let n_{gb} as the false negative predictions that predict the true gate pixels to be backgrounds. Then $t_g = n_{gg} + n_{gb}$ will be the total number of gate pixels. Mean IoU is defined as $n_{gg}/(t_g + n_{bg} - n_{gg})$, where n_{bg} is the number of false positive predictions. Similarly, pixel prediction accuracy can be defined as n_{gg}/t_g.

2) Inference Speed: To successfully finish a vision-based autonomous drone racing task, in addition to achieve high prediction accuracy, it is even more important that the gate recognition can be executed at a very high frequency in order to generate control signals. As a result, we also evaluated the inference speed of FCNs by calculating the average inference time over the test set. Results can be found in Table I.

D. Implementation Details

We implemented our model design and executed all experiments using PyTorch on Google Colab. Both training and evaluation experiments including inference speed calculation is executed on a NVIDIA Tesla P100-PCIe 1190 MHz GPU with a graphic memory size of 16GB.

V. Results

In Table I we listed the 9 FCN architectures corresponding with number of layers and parameters, memory storage and inference speeds. Owning to the relatively more concise structure, VGG5 based FCNs consume less memory and can operate at a higher speed than VGG11 based FCNs.

TABLE I
Variant FCN architectures

FCN Architectures	*Conv. Layers*	*Number of parameters*	*Memory Storage*	*Inference Time*
VGG5-32s	6	3.92M	**190.23 MB**	**1.144 ms**
VGG11-32s	9	9.23M	243.39 MB	1.385 ms
VGG11-32s-BN	9	9.23M	337.06 MB	2.166 ms
VGG5-16s	7	3.91M	**190.22 MB**	**1.385 ms**
VGG11-16s	10	9.22M	243.38 MB	1.455 ms
VGG11-16s-BN	10	9.23M	337.06 MB	2.271 ms
VGG5-8s	8	3.91M	**190.24 MB**	**1.386 ms**
VGG11-8s	11	9.22M	243.40 MB	1.725 ms
VGG11-8s-BN	11	9.23M	337.08 MB	2.440 ms

The loss curves during training process for different FCN architectures are shown in Fig. 2. Each loss curve is smoothed by a exponential weighted moving average function with weight decay 0.95 for better demonstration. It can be seen that while the FCN-32s with no skip-layer structures can only converge to a loss value of nearly 0.1, FCN-16s and FCN-8s are able to achieve smaller loss value, in which VGG11 based FCN-8s with BN ended up with the smallest BCE error. Adding BN can substantially accelerate learning at the initial stage, but the long-term benefit is insignificant. There are also trivial differences among different VGG models given the same skip-layer structure. In contrast, modifying skip-layer structures can largely affect the converged loss value.

In Fig. 3, the mean IoU and mean pixel accuracy evaluated over test set is demonstrated. In consistent with the smaller loss during training, FCN-16s and FCN-8s also outperform FCN-32s in prediction. Besides, FCN-8s slightly performs better than FCN-16s by connecting finer feature maps with decoder. More importantly, while operating at a higher speed, our VGG5 based FCNs can achieve the equivalent performances in prediction accuracy as VGG11 based FCNs, which make it the most suitable model to run on an embedded system on real drones.

Finally, we demonstrate the essential role of skip-layer structure in predicting gates accurately in Fig. 4. We randomly sample the original image and mask from test set along with predictions from FCNs. FCN-32s predictions are blurred due to the coarse stride in upsampling, while FCN-16s can detect the gate edges much more in detail and FCN-8s performs the best.

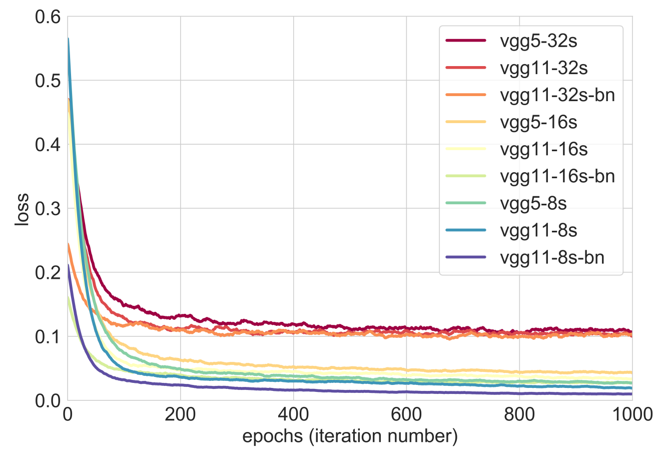

Fig. 2. Training loss of FCNs with variant VGG architectures and skip-layer structures.

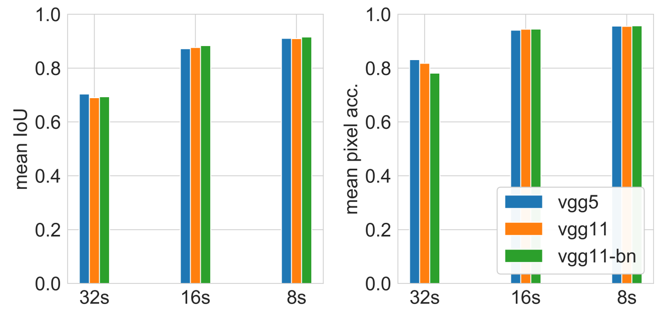

Fig. 3. Mean IoU and pixel prediction comparison of FCNs evaluated on test set. While modifying VGG architectures has slight effect on the metrics, introducing skip-layer structures can highly improve the prediction accuracy.

VI. Conclusion

In this paper, a FCN-based semantic segmentation neural network model to recognize gates in drone racing is formulated. It can be shown from the experiments that skip-layer structure plays an essential role in making accurate predictions especially for small gates by focusing on local areas while respecting the global structure. Our proposed VGG5 based

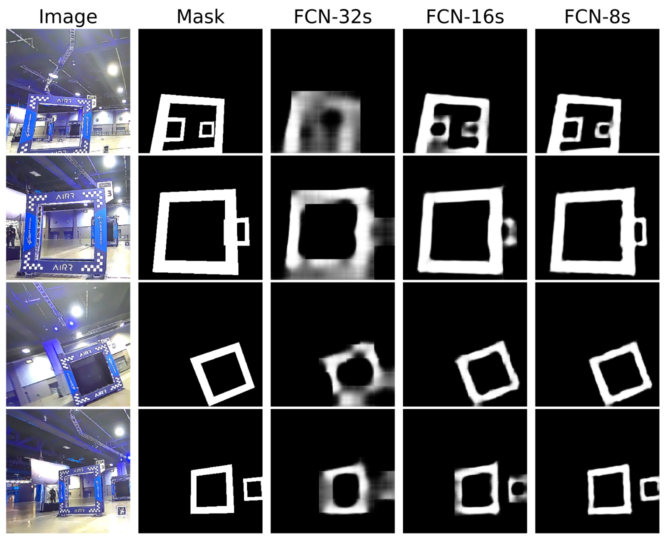

Fig. 4. Demo for predictions of FCNs with different skip-layer structures on random inputs from test set. Refining fully convolutional neural networks by fusing information from layers with different strides improves segmentation detail and accuracy.

FCN-8s not only achieves a equivalent performance as more complicated networks such as VGG11 based FCNs but also operates at a higher speed. However, due to hardware limitations, we can't test our proposed model in real embedded AI computing systems such as NVIDIA TX2. Future work can be done both on applying the algorithm to a real drones and further exploring new network structures to save computation resources and recognize gates more accurately.

References

[1] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.
[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#SimonyanZ14a
[4] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2016, cite arxiv:1603.07285. [Online]. Available: http://arxiv.org/abs/1603.07285
[5] "AIRR - The drone racing league," https://thedroneracingleague.com/airr/.
[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167
[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." in *ICLR (Poster)*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14