Model – final score 0.49256118836

Final model had a tricky architecture. It is tough to explain with words, so I draw the picture in Appendix A. Basic principle is to create system where layers passed along the network to learn nice distinctions within the objects of the scene.

Such a deep model with some of information is being passed all the way through the network allows to catch small details, which is required to detect target from bigger distance. And that was primary issue of the project.

Addressing review questions.

Why are there 2 sets of encoders/decoders?

That architecture proved itself useful in the cancer research, but ultimately idea is to get better chance of being correct. Deeper networks enable learning of small details and in this project main challenge was to identify target from far, when it is just a few pixels big.

In addition, when we have 2 encoder/decoders they are unlikely to do the same mistake and this result in more powerful learning (correct each other).

Why that special skip connection?

Big problem with deep architectures is the loss of the signal. When we make picture smaller we lose some information and second encoder/decoder may not have much to deal with. So in order to fight it we pass early layer and combine it with heavily processed layers. That enhance signal and allow final decoder to be significantly more powerful.

What does adding more layers to your model do?

Allow the model to learn smaller details about the image.

Why have you chosen that structure?

I do not have smart answer here. Because I tried, a bunch of different ways and this one worked the best with my benchmarks. Multiple nets got me to the same loss, but this one did it faster. So when I started full training and collecting the data I just used that.

In order to play with different sizes of nets I made a function to create encoder/decoder of select size (encode_decode in model_training.py)

Dimensions of layers.

Calculation of number if filters seems like a waste of time. You just

Layer	Depth	Shape
Inputs	3	160 x 160
Encoder 1	64	80 x 80
Encoder 2	128	40 x 40
Encoder 2.1	254	40 x 40
Encoder 3	254	20 x 20

Encoder 4	512	10 x 10
Fully connected	1024	10 x 10
Decoder 1	512	20 x 20
Decoder 2	254	40 x 40
Decoder 3	128	80 x 80
Decoder 4	64	160 x 160

This is standard approach to increase depth as we decrease height and width (and we do with stride 2). Then we go another way to get back to 3 dimensional RGB image.

You can calculate actual number of filters here.

Issues

Provided dataset did not include enough training examples that contain target hero, so model could not converge. Later explorations showed that it is better to neglect provided dataset and collect own images.

Adam optimizer has rapid gradient descend, so learning rate of 0.0005 was used to fight it. Otherwise, model stuck on local minimum.

Data

Total amount of collected images is around 3000. In addition, all of the pictures and masks are flipped horizontally in order to increase dataset. My estimation is – that architecture is space specific; flipping image change where the object is. As the result, it is beneficial to do so.

Extensions

Model can be retrained to follow different target and with its current state it would take significantly less time and effort, as pre-trained model is already knows well the color pallet of the simulator and ways to deconstruct image. However, in order to retrain model to follow new object at least 1000 of new images of new target within the environment would be needed.

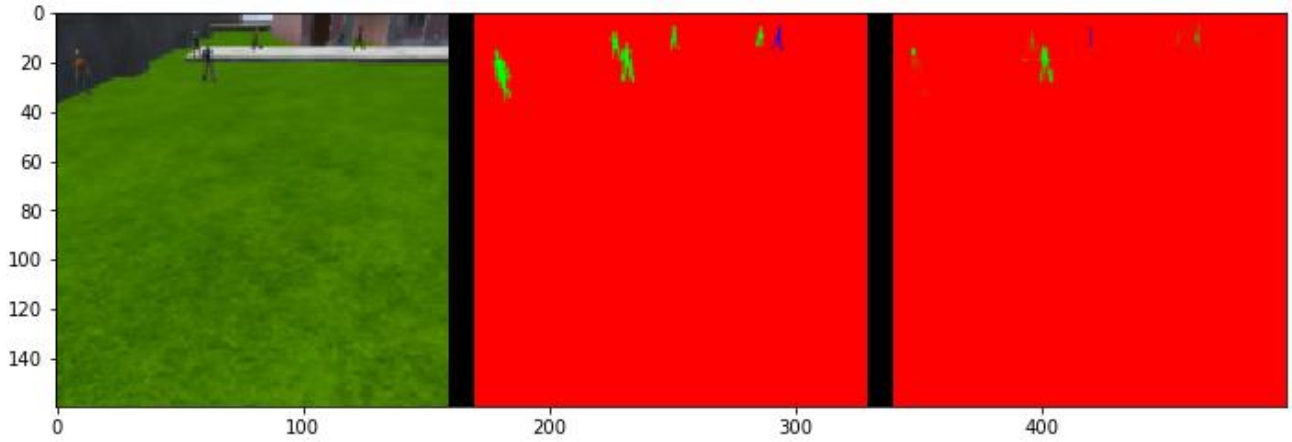
However, the network is powerful and can be scaled to include additional classes.

Addressing review.

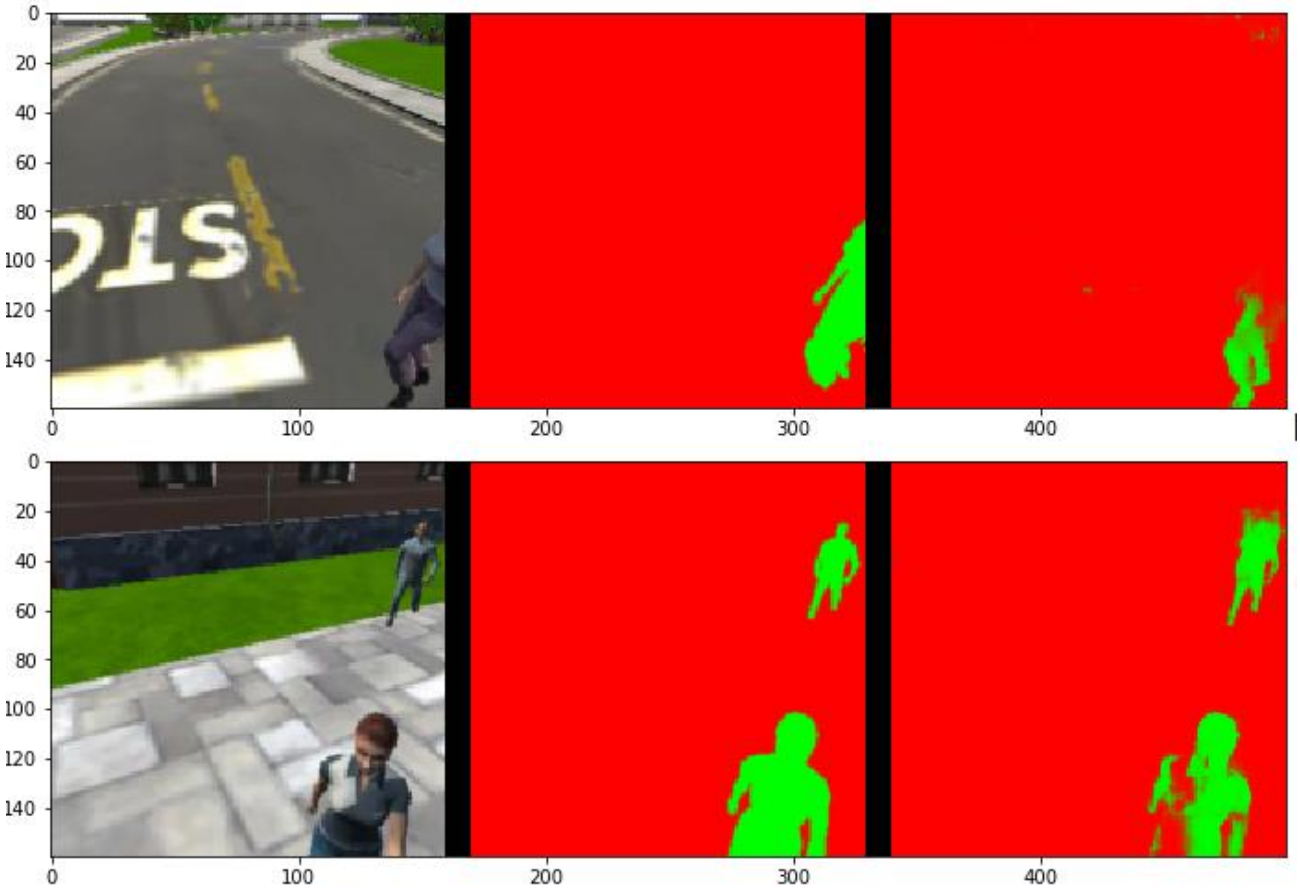
This however, isn't true for objects other than humans. The model has only been fed data of humans and has learnt to distinguish between one human, in red clothes, from the other humans in this simulation. It would need a variety of different classes fed to it such that the model develop to recognise general features well. For eg, when using transfer learning, models that are chosen are pre-trained on imagenet that contains 1.2M images and a 1000 classes.

To make this model recognize, say, a car, what changes would be required? Would the model structure need to be changed?

I have to disagree. In case of the real world, it would probably be true, but we have to remember that huge part of this project is detection of the objects from the distance. As we can see from the pic bellow, we get close to no information about the shape.



To prove that it is all about the color we can also see what kind of mistakes network makes when it comes to people up close.



Lack of colors in simulated environment make network that cares more about color pallet, then shape.

As the result, I still stand that if you take this trained network, introduced 1000 images where new target would be a car or a dog or whatever, you will be able to retrain the network for new target. Of course if we are talking about the same simulator.

Not sure what to say about "what changes would be required". There is no such thing as architectures that good to detect people or car. In deep learning, we are programming with data (heuristics), so we need new data. Pictures from the drone with something else marked as blue in this case.

Hyperparameters

- ➔ **Epoch** – is set to 20, but model was retrained multiple times. In total about 200.
- ➔ **Batch size** – 4 which along with small learning rate decrease rate of gradient descend. It might sound like a small batch number, but if you really calculate, it takes almost 15 gig of VRam when training this model. Had to fit it on two GPUs.
- ➔ **Learning rate** – 0.0005 as stated above, to fight sharp gradient of the Adam
- ➔ **Steps per epoch** – I used recommended formula "number of training samples" / "batch size"

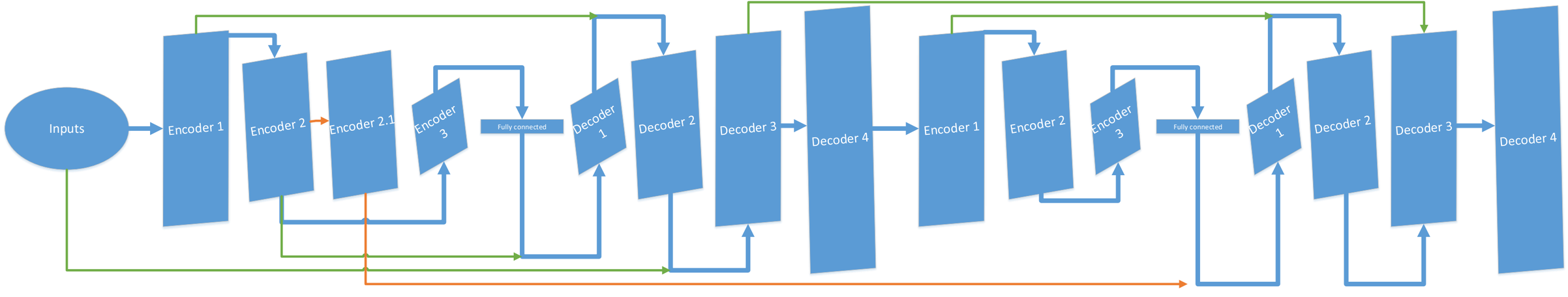
The student is demonstrates a clear understanding of 1 by 1 convolutions and where/when/how it should be used.

I have trouble understanding what exactly to put in here. 1 by 1 convolution allows to preserve original tensor height and width, while increasing the depth. That is what we love about Dense layers. However, since it is convolution, spacial information is preserved, which is our goal.

One by 1 means that filter is 1 pixel in both height and width.

Appendix A model

- ➔ **Blue lines –** direct connections (feed forward)
- ➔ **Green lines –** layers that are getting passed through to concatenate with following layers
- ➔ **Red line –** special case, layer is passed to second decoder



**** there is encoder 4 missing, but it will just make graph longer, imagine encoder 3 is just connected to 4***

***** I do understand that I can generate picture from the model, but in this case, it is unusable mess***