The proposed challenge is to segment the sky pixels in a given image.In other words, given an input image the task is to propose an algorithm that returns a binary mask of the same size as the image, indicating if each of the pixels belongs to sky or not. The task is related to a couple of standard challenges in computer vision viz. Image matting and semantic segmentation. As part of the solution, I have proposed algorithms/models that belong to one of these fields applied to the task of sky segmentation. As part of the rest of the report, we will first look at the datasets used and the way we can get them. We will then look briefly at three different deep learning models that help us to solve the task at hand. We will analyse the results, both qualitatively and quantitatively before proceeding towards proposing the next steps to improve the solution further. As proposed in the challenge description, the purpose of identifying the sky mask is to be able to add animations; We will look at a possible semi-automatic pipeline that helps us achieve this.

We used the following two datasets for training and evaluating our models - ADE20k and Cityscapes. These datasets are generally used for semantic segmentation with different classes. For example, Cityscape dataset provides images of streets and roads labelling people, vehicles, trees, sky etc. It is generally used in the context of autonomous driving for detecting people and other vehicles on the road. Since these datasets are not specific to sky segmentation, I had to create my dataset from these sources by labelling all other classes as one and sky as the other class. Hence I created a dataset for binary classification of each pixel in the image (segmentation). For creating these datasets, I utilised the scripts provided by the authors of the above datasets. For example, Cityscape dataset provides scripts to prepare the dataset. And for the other dataset, we used the dataset created by Liba et al. that is provided here. Both the datasets have the original images and the corresponding masks for training deep learning models. There is also another interesting dataset - SkyFinder, which contains images in all weather conditions from static cameras. I did not use this dataset, but it could be an important next step to use these images because of the variety it provides.

Now, we will briefly look at the different deep learning models that have been used to perform the segmentation task.

1. Fully convolutional network (FCN): Convolutional neural networks are used in the image recognition tasks where you assign a class to an image. The penultimate layers of an CNN are fully connected layers leading to a final layer that has the size of the number of output classes. Long et al. proposed to replace the final fully connected layers with convolutional layers and with a combination of upsampling, we can obtain pixel wise classification instead of image level classification. We use a pretrained (on ImageNet) resnet model and replace the last layers with convolutional layers for our task.
2. U-Net: This is also a fully convolutional network that does not contain any dense fully connected layers. The architecture of this network is U-shaped, with the left side being an encoder with convolutional and max pooling blocks and the right side with transposed

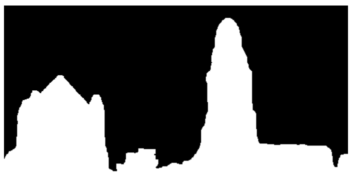
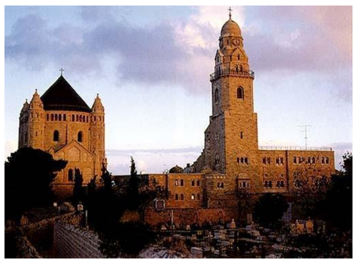
convolutional layers that enable pixel-wise localisation. It was initially proposed for biomedical image segmentation and has been used for various other image types afterwards. We use the original UNet model and train in on the data we collected.

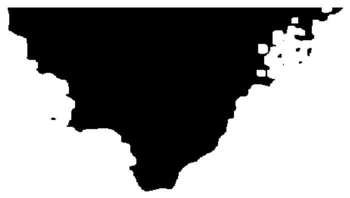
3. U2-Net: This model was initially proposed for salient object detection, which is an extension image matting task. As the name indicates, the architecture of the model is a nested U-structure. The authors utilise techniques like multi-level and multi-scale feature integration in order to capture the contextual features in an image. They extend the U-block by adding residual connections. Unlike the first FCN, UNet and U2Net do not have any backbone and are trained from scratch. We use the original U2Net model for training and evaluation.

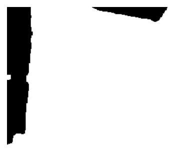
We trained these networks on the ADE20k dataset with the following parameters:
Batch size = 12
No. of epochs = 50
Adam optimiser with a starting learning rate of 1e-4
These parameters along with the choice of the model can be changed in a config file provided in the repository. For evaluation, we used the following metrics - mean absolute error and mean fscore. We evaluated our models on two datasets - validation set of ADE20k and Cityscape. The table below gives the average scores of the two metrics. Observing the scores in the table, we can see that the first FCN model performs well on both the datasets. It is also interesting to note that it has a better score on the Cityscape dataset, though it was trained on ADE20k.

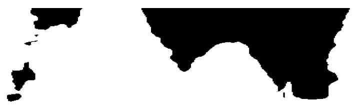
	FCN		UNet		U2Net	
	MAE	F-Score	MAE	F-Score	MAE	F-Score
ADE20k	0.017 (0.02)	0.98 (0.01)	0.13 (0.13)	0.88 (0.13)	0.01 (0.01)	0.98 (0.01)
Cityscape	0.01 (0.01)	0.99 (0.007)	0.19 (0.09)	0.84 (0.06)	0.09 (0.09)	0.94 (0.06)

Additionally, please find some qualitative results here from both the datasets for FCN:

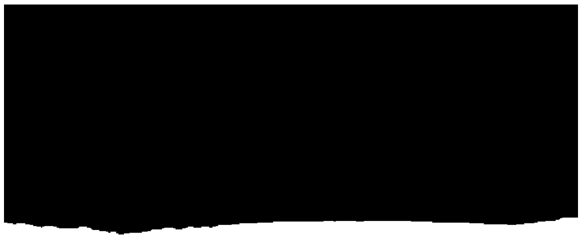



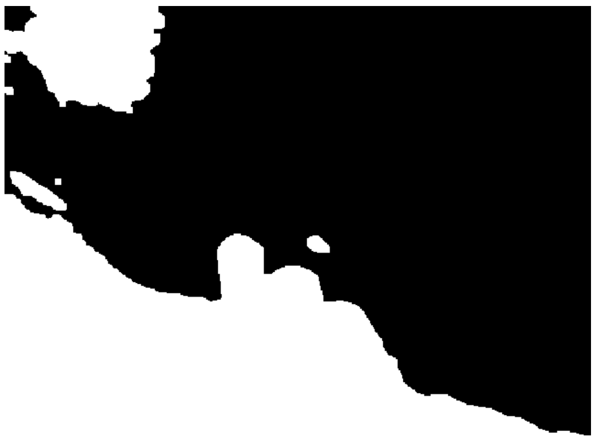


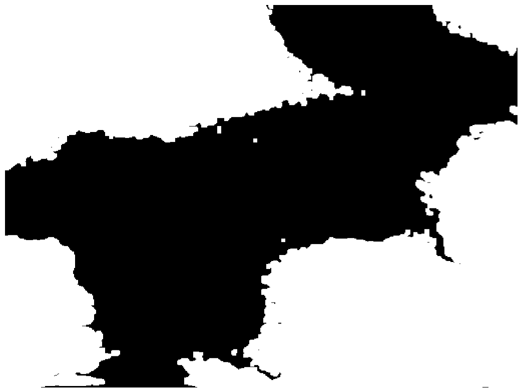




In addition to the above test images, we also ran the model on some images from the internet. These pictures contain skiing, skydiving, night sky etc. A reasonable performance was seen on most images, but the night sky was a challenge. Please see some of the results here.







Considering that the night sky is a challenge, we can follow the approach and post-processing proposed by Liba et al.

Now, we will describe a few steps that could form the future work in this task and some of these are the steps I would have followed if I had more time and computing resources.

- Train the best performing model on a combination of different datasets and hence improve the variety of training data that the model can learn from. Also, use other metrics stated in the above papers, especially mean IOU.
- Tune the different hyperparameters like batch size, number of epochs, learning rate for the specific combination of model and dataset. Now, we are using standard values across the three models. In addition, I would have liked to do better exception handling and unit testing in the code.
- The post-processing steps used to refine the output of the neural network is rudimentary involving a gaussian blur and other morphological operations. I believe the results would improve if we use more sophisticated approaches as proposed in Qin et al. or Liba et al.
- The inference speed is low for these models as it takes a couple of minutes for 100 images in the test set. For an application running on mobile, this can be a deal-breaker. I would optimise the model for size and inference speed using techniques like pruning or quantisation. One of the challenges of using these techniques is that we need to perform a trade-off between model size and accuracy.
- If we can improve the inference speed, we can look creating an ensemble of models so that we can get the best of all three models, hopefully improving the final segmentation.
- As the goal is for the users to use this model to automatically create animations by replacing the sky pixels, we can explore different ways for the user to correct the predicted mask. This way, we can also gather feedback to improve our training algorithm.
- For now, we did not investigate how the model works on images with no sky (e.g., indoor images). One of the next steps could be to collect such images and explore how the models perform on such data.
- For the purpose of automating the addition of animations, we could think of the following sequence of operations, inspired by Zou.
    - Segment the sky pixels using the trained model
    - Provide a way to fine-tune the mask to the user
    - Using motion estimation, image relighting and blending, create an animated video.