SkateMAE: Skateboard Pose Estimation with Synthetic Data

Yau-Meng Wong (ywongar)

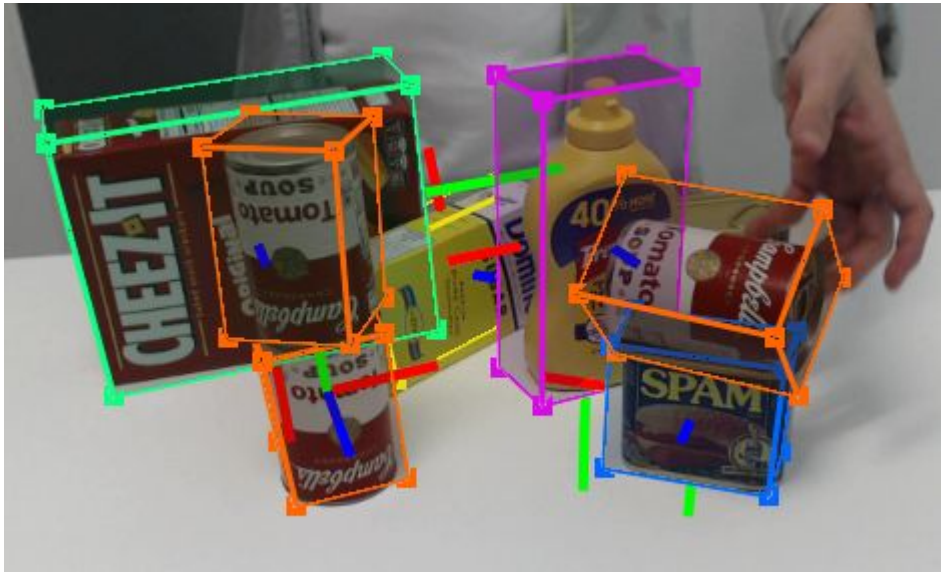
Why try skateboard pose estimation?

- Application of AI to sports is a growing field
  - See Google's TacticAI for Liverpool FC
- Skateboarding is a popular sport (was in the last Olympics)
- I like skateboarding
- Skateboarding tricks (on flat ground) are partially defined by the board's rotation, so pose estimation can help classify tricks.
  - This is the end goal of this project
  - For now will only focus on board rotation, because translation is not important in trick classification

Why try skateboard pose estimation?

Traditional Object Pose Estimation (Supervised)

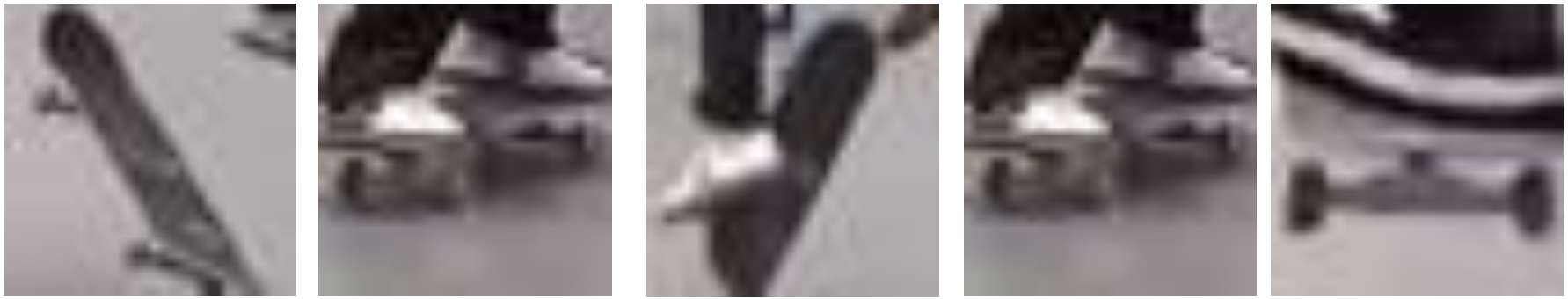
- Object pose estimation is difficult without ground truth data to train on.
- Typical datasets are hand labelled or created in controlled environments
- No such dataset exists for skateboards



Creating a dataset for skateboard pose estimation

First collected real world data to test on

- Downloaded and processed skate videos from Youtube
  - Specifically from the Battle At The Berrics series from the Berrics Youtube channel
- Used MaskFormer pretrained on COCO to get bounding boxes of the skateboard.
- Cropped the image to just the square bounding box, resized to 32x32



Creating a dataset for skateboard pose estimation

Example original video frame with bounding box diagonal (red line).

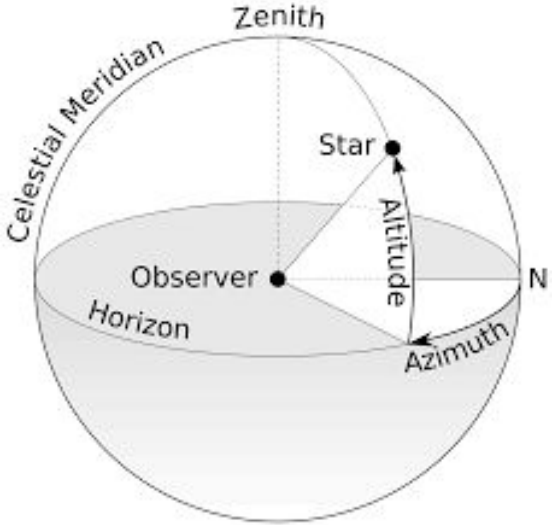
Collecting synthetic data

The real world data was unlabelled, so I had to collect some synthetic labelled data

- Downloaded a free 3D model of a skateboard

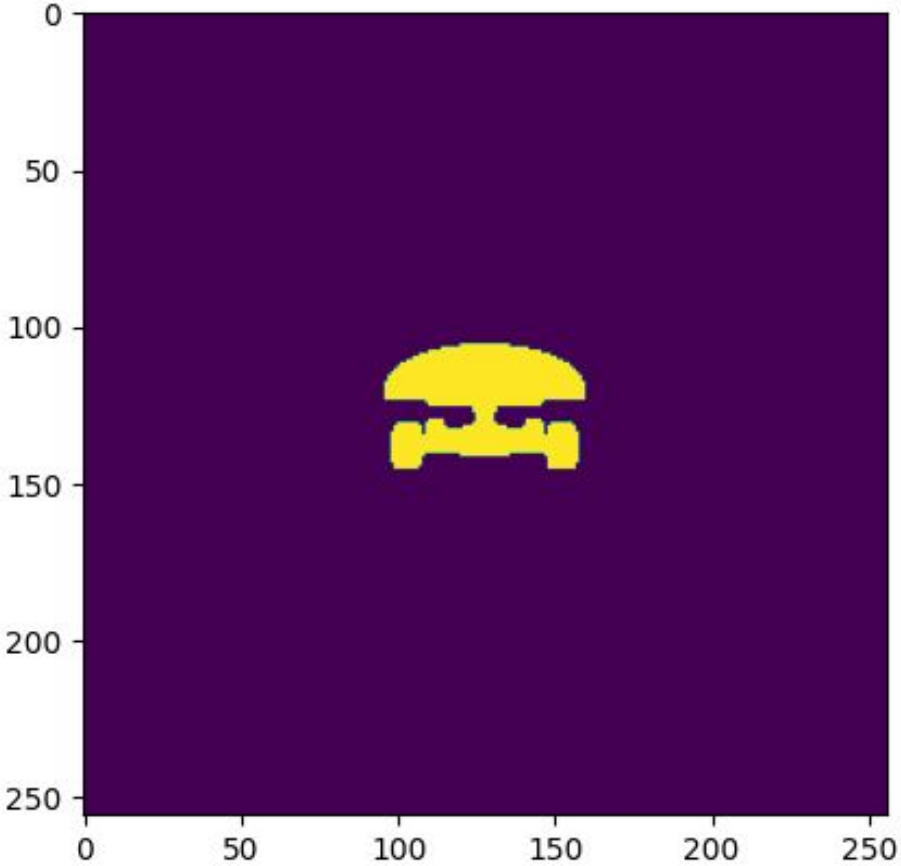
Collecting synthetic data

- Rendered the model with Pytorch3D at randomized camera angles
- Used an FOV Camera, so the “ground truth” values saved were distance, elevation/altitude, and azimuth of the camera.



Collecting synthetic data

- Used the “A” channel from the rendered RGBA image (i.e. object opacity) as a mask to overlay the skateboard with randomized crops from the real world data.
- Resized to 32x32 to match real world data.



Collecting synthetic data

- Final synthetic images







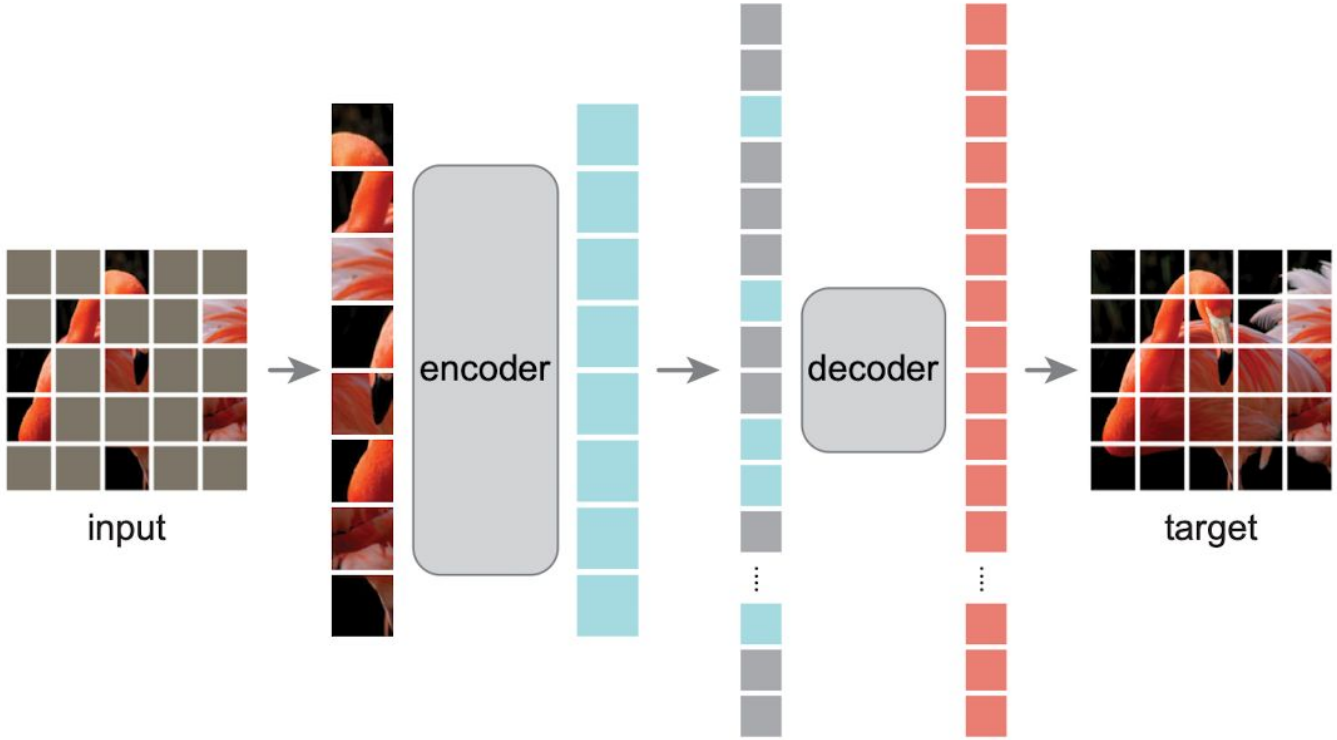

SkateMAE (Masked Autoencoder)

- Pretrained the encoder/decoder on image reconstruction with a combination of real and synthetic data.
- Then trained the encoder + 3 MLP classification heads on the synthetic data to predict distance/elevation/azimuth of camera.
- Used mean squared error loss because all three of these are continuous variables so closer predictions should be rewarded.

(forgot to re-normalize)

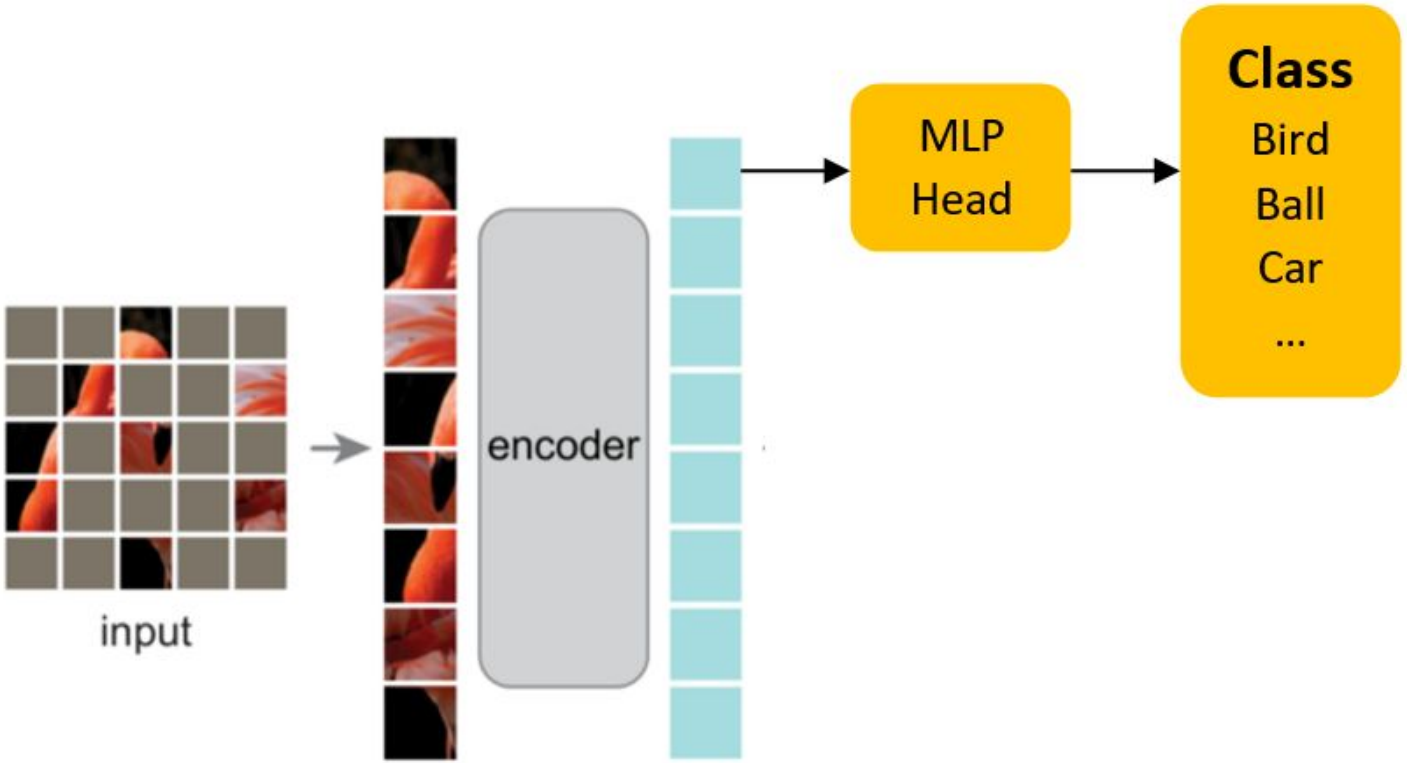
SkateMAE

Pretraining (2k real + 30k synthetic images)



SkateMAE

Training (30k synthetic images with ground truth camera angle labels)



Initial Results (low res render bc I ran out of colab gpu)

Real world video

Predicted skateboard pose

Issues with Sim2Real

The real world data has much more variation than the synthetic data. For example there are no legs/feet in the synthetic images.

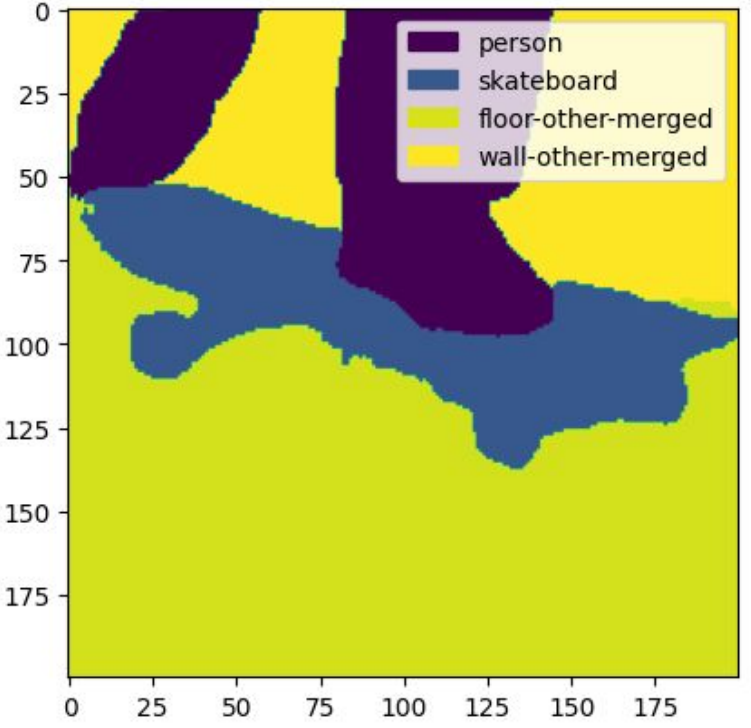
All of this makes it hard for the model to generalize to the real data after being trained on the synthetic.

Real:

Synthetic:

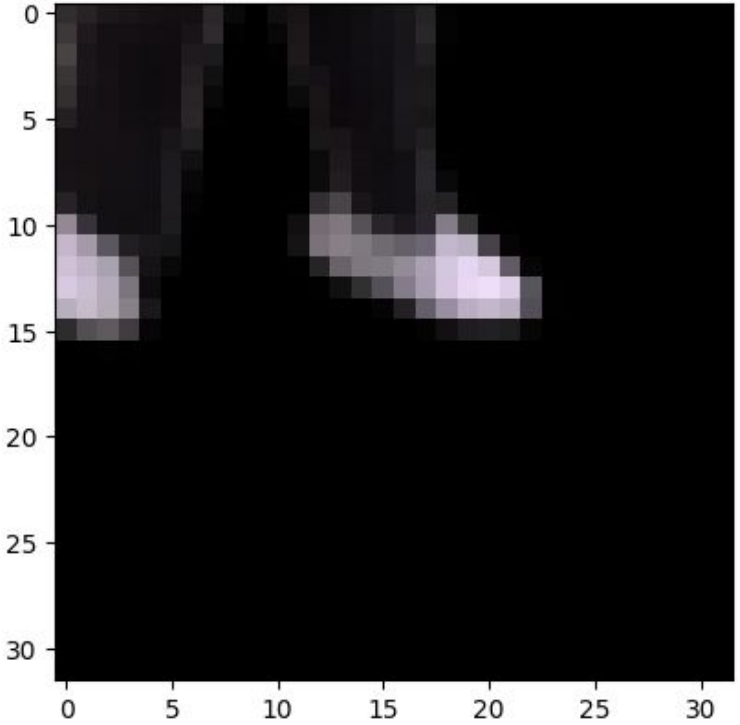
Reducing semantic difference between real/synthetic

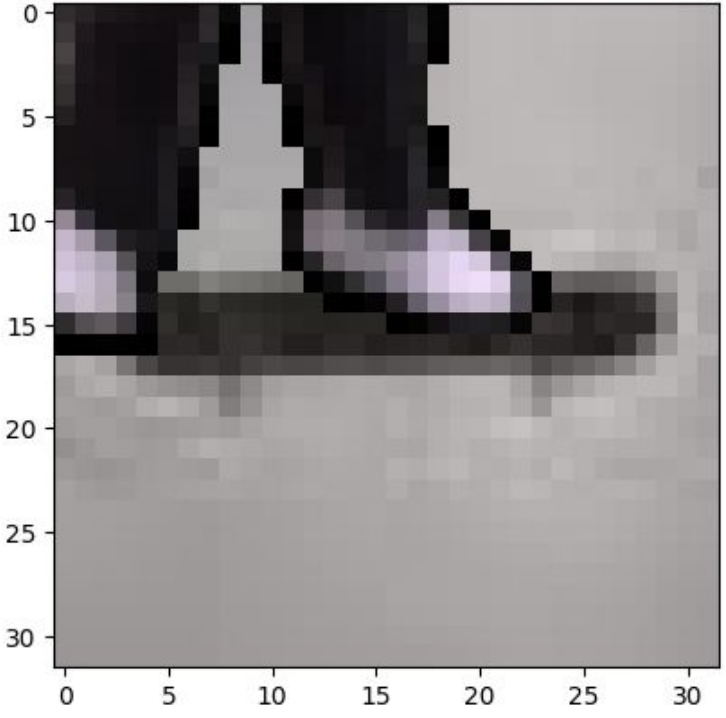
Used pre trained semantic segmentation MaskFormer to segment real video frames



Reducing semantic difference between real/synthetic

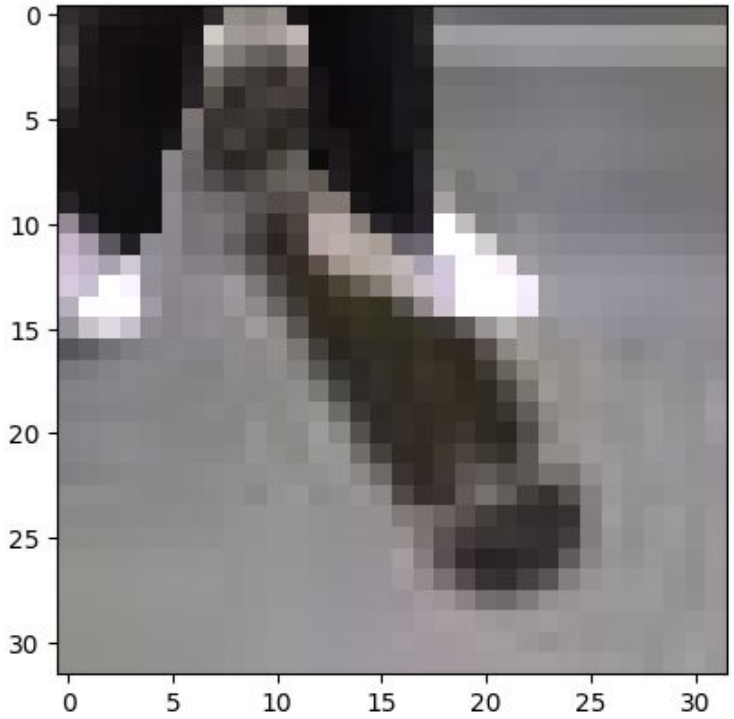
Created “leg masks” to overlay on synthetic images (resized to 32x32)

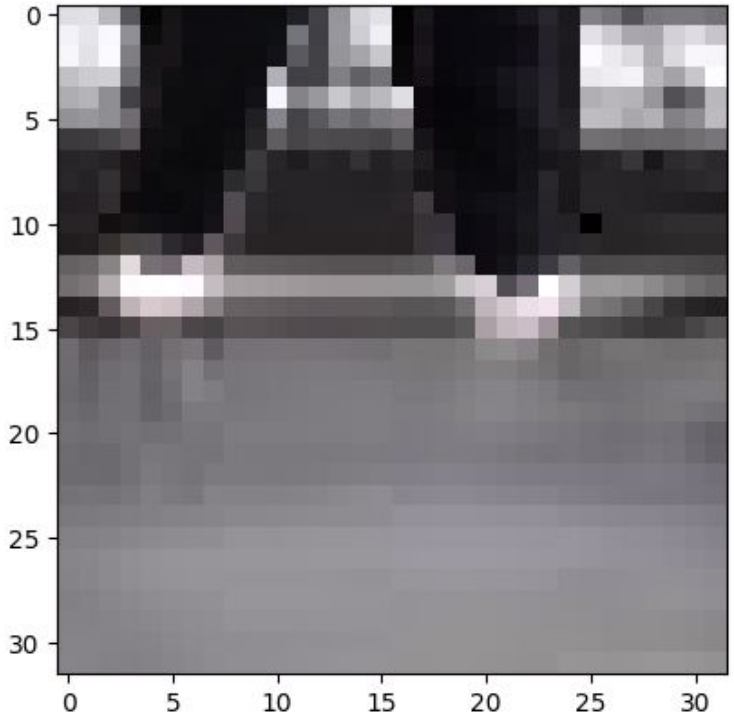




Reducing semantic difference between real/synthetic

Cleaned the mask up using binary erosion with a convolutional kernel. Final results:





Reducing semantic difference between real/synthetic

Stacked “Add Legs” data augmentation on top of random color jitter and image translations.

Overall this slightly improved test accuracy (on the test set of synthetic data)

But how does it transfer to real world data?

Current Results

Real world video

Predicted skateboard pose

Compare pre vs. post data augmentation

Real world video

Pre data augmentation

Post data augmentation

Future directions

Right now camera angle is being predicted frame by frame. Including context from previous frames could improve stability

Will need to redo the synthetic generation to mimic a continuous video (generate images sequentially with restricted angle changing frame-to-frame). Cuts between different shots in real world video seems tricky to deal with.

Perhaps try a non MAE method? Was considering a GAN at first.

Higher res images? Perhaps the low resolution is making the pose estimation task more difficult

Summary

- Created real and synthetic dataset for skateboard pose estimation utilizing MaskFormer for bounding boxes + semantic segmentation and Pytorch3D for synthetic data generation
- Trained MAE, got bad results
- Added data augmentation to bridge sim2real semantic gap
- Trained MAE again, got slightly better (?) results