Semantic Segregation

what is semantic segregation for?

Semantic segregation is used to show if a pixel belongs to a given class. I chose to do this topic because it was from my opinion the most interesting one. I used the *Pascal VOC 2012* dataset for this, because I thought it'l lead to better results, than the *CUB200-2011* dataset. The pascal vod dataset consists of 20 classes (or 21 with the blank one)

How can you find out the classes

I have a script *colormap.py* which gets all the classes in an array and saves this as file *classArray* (The classes are encoded in the pixel rgb value).

main.py

After getting the classes stored in the file I used the other python script *main.py* for doing the semantic segregation. the libs I used for this are:

- segmentation_models, Unet
- segmentation_models.backbones, get_preprocessing
- segmentation_models.losses, bce_jaccard_loss
- segmentation_models.metrics, iou_score
- PIL, Image
- numpy
- matplotlib import pyplot
- matplotlib.image, imread
- import pickle

First the data gets loaded from the dataset, as well as the classes (the pixesls). Afterwards the X_train,Y_train,x_val,Y_val are getting created. For the lib., the Y variables have to be the pixels with class labels instead of the rgb. therefore they are getting processed by the "fastApproach" function. After some other preprocessing like normalizing the pixel value the data is getting fed into the model.

Problems I faced

1. I just habe a surface so it takes for ages to train, so I reduced the batchsize to one
2. the dataset is bigger than my whole memory, so I just took 200 samples
3. the pixel encoding was just weird
4. It reached an accuracy of 98% with the samples, before my computer can't handle it anymore

PCA t-sne.py

what is this for

Datasets have a lot of variables and they somehow have to be computed down to a lower dimmension to visualize without a lot of loss in data. Therefore we need dimensionality reduction (PCA t-sne)

PCA is reducing the dimmensions by calculating the eigenvectors as well as the eigenvalues and coosing the minimum one. So it tries to provide a minimum of variables with the most information, how the original dataset was distributed.

T-Distributed Stochastic Neighbouring Entities (t-SNE) t-Distributed stochastic neighbor embedding (t-SNE) minimizes the divergence between two distributions, but t-SNE is quite slow. So it's recoommended, that the dimmension gets first reduced with e.g. PCA and afterwards t-sne is used.

my Results

PCA doesn't really show the differences in the date, however t-sne does this quite good. But it works best by combinding both algorithms for t-sne and PCA (Picture is provided on the poster)

how I did it

I used the minst dataset for getting the datasets seperated. Because I thought numbers might be the best to distinguish to have some good results. I also used some libraries:

- numpy (standard library for numerical operations)
- pandas (used for plotting)
- sklearn import datasets (I used the mnst dataset from sklearn)
- sklearn.decomposition, PCA (I used the PCA from sklearn)
- sklearn.manifold, TSNE (I used the T-SNE from sklearn)
- matplotlib.pyplot (also for plotting)
- mpl_toolkits.mplot3d/ Axes3D (also for plotting)
- seaborn (for the color while plotting)

Image recunstruction

It's used to reconstruct missing parts of an image. The overall structure of it is an autoencoder. We used several methods and looked, how good they are. E.g. AlexNet, VGG, ResNet etc. we used pretrained nets. The Convolution and maxpooling layer is replaced by an Convolution Transpose and upsample. For resnet, the resblock is kept. The resblock, which shrinks the feature map size is replaced by upsample. All to see in our poster.

Image Classification

in image classification we observed, how diferent training parameters such as learning rate and dropout are affecing the accuracy of the model. In the second experiment we compared different models to see the different accuracy. (for VGG16, ResNet50, DenseNet121) VGG16 took way to long to train, because of its complex structure. Resnet is just 50 layers deep, resudual training allowed us to train networks with many layers by skipping connections between them. DenseNet121 used a lot of memory compared to resnet, but it required less time and less computation MobilenetV2 mobile optimized net, which uses the resedual structure to skip layers and decrease complexity. The testing results can be seen on the poster.