Computational Intelligence Lab: Road Segmentation Project

Clément Sicard, Kajetan Pyszkowski, Alexis Tabin, Fayçal M'Hamdi
Department of Computer Science, ETH Zurich, Switzerland
Group: IciCaOverfit

***Abstract*—From edge-detection segmentation to segmentation based on clustering, researchers around the world made many advancements in the previous years concerning the task of segmenting roads from aerial images. Road Segmentation from remote sensing image analysis plays an important role in automating operations such as urban planning, autonomous driving, map definition... This paper describes a method leveraging the power of visual transformers, and combining them with U-shape networks such as UNet, in order to improve performance on road segmentation tasks. This method allowed us to reach a final score of 0.91908 for the Kaggle Competition of the course.**

***Index Terms*—Computer Vision, Road Segmentation, Visual Transformers, U-shaped Networks, Swin Transformer, Ensemble Methods**

I. Introduction

Due to urbanization, nearly a million kilometres of roads are created every year. This phenomenon leads to a constant change of information needed by many industries, such as autonomous driving and urban planning. Map creation is an expensive process, both in computation power, with traditional methods, and in cost.
As manual techniques to update road locations are no longer an option, the advancement of deep learning helped tackle this problem. Such methods are, among others, region-based segmentation, edge-detection segmentation, clustering approaches, or even methods using weakly-supervised learning in CNN. [1].

In this work, we implemented, in parallel with more traditional techniques, a new method using one of the latest innovations in machine learning: visual transformers. Given aerial road images, our method aims to extract the position of roads on them. To do so, at inference time, our trained model divides each image into smaller patches of pixels of equal size and then assigns a label to each of them (0 if there are mainly no roads in this portion, 1 otherwise). We then regroup all the results for these smaller patches into a mask of the same size as the image.

Due to the low amount of available data, our first goal was to augment the dataset by applying transformations. As pixels labelled as roads were in a clear minority compared to others, our second goal was to handle class imbalance by setting an appropriate and sometimes infrequently seen loss function. Finally, three baseline methods were used to compare performance with our proposed model.

II. Related Work

In this section, we did an overview of what has already been done in the field of road extraction from aerial images. We reviewed different techniques like classical CNN, auto-encoders, vision transformers, and multi-scale transformers.

A. CNN Architecture for road extraction

CNNs are widely used in the field of image classification. For example, CNN [2], [3] are capable of capturing precise road feature representation. We also noticed that CNNs [4], [5] with an encoder-decoder structure were already performing well for remote sensing image segmentation. Those were then further improved by road structure refined CNN (RSRCNN) [6] that considers both spatial correlation and geometric information of road structure.

B. Vision and Swin Transformer

Transformers have proven to achieve outstanding results in numerous NLP tasks [7], [8], [9], [10]. The pioneer ViT [11] introduced a novelty based on Transformer which manage to achieve state-of-the-art performance on image classification tasks. The drawback with transformers is that they are extremely needy in data. For this reason, Swin Transformers [12] were introduced. This hierarchical Transformer used Shifted Windows to compute its representation. It managed to effectively strengthen the information interaction between patches. Patches with a fixed scale were limiting further model performance improvement, this was the reason to introduce a dual-resolution road segmentation network (DCS-TransUperNet) [13]. Design based on CSwin Transformer, they introduced a new FFM (Fusion Feature Module) [14] to build enhanced feature representation with global dependencies, using different scale features from subnetwork encoders.

III. Methodology

A. Data Preprocessing

1) Initial Dataset: For this task, the provided dataset consists of 144 samples, from which 134 were used for training and 10 for validation. A sample is a pair of a (400×400) image and a ground truth mask, in which a pixel value of 0.0 means that the corresponding pixel in the image is not a road, and a pixel value of 255.0 means that the corresponding pixel in the image is indeed a road.

2) Submission Format: We divide each image into (16×16) non-overlapping patches, and the submission mask for each image consists of a CSV file with one-row value per patch. This value is determined by computing an average value on the predicted mask for each patch, and rounding it to the closest value in $\{0, 1\}$ based on a certain threshold $\alpha \in \{0, 1\}$.

3) Dataset Augmentation: At this point, as one could guess, the original training set is presumably not populated enough to train a deep neural network well. A straightforward way to increase its cardinality is to apply different kinds of transformations to the images. We applied ± 90 degrees rotations, horizontal and vertical flips, and we also include the regular image. Each of which is simultaneously done to the image and to the mask to maintain coherence between both. (Fig. 1).

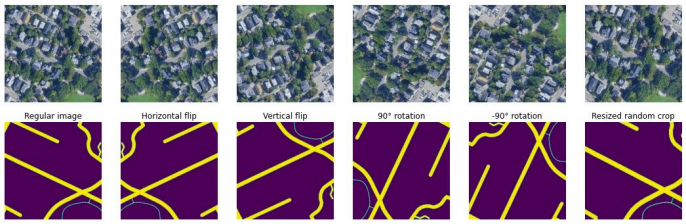

Fig. 1. Illustration of different transformations

Furthermore, as the pre-trained Swin Transformer from PyTorch is initially trained for (224×224) images, often we resized and/or cropped our images to (208×208) – the closest multiple of 16, so that the crops would the overlap the least possible when joined together to reconstruct the image but we would still be able to compute patch dependent metrics. This process would scale our dataset by a factor of 4 (4 crops each anchored into one of the corners' image). (Fig. 2).

From there, we could further augment the dataset in this case, by then applying all or any of the 6 transformations in Fig. 1 to the cropped images.

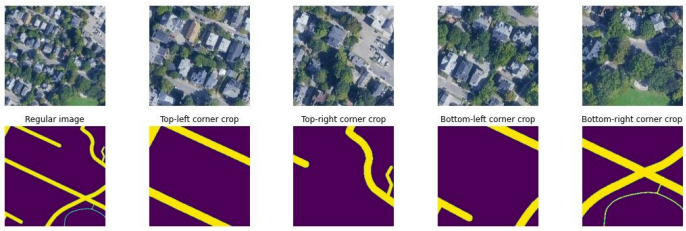

Fig. 2. Illustration of our corner-cropping method

These transformations are part of our dataset, that we query by batch using PyTorch's `DataLoader` class on top of our dataset. All the transformations use methods defined in the `torchvision` library for PyTorch [15]. By using these transformations and cropping, we are now able to feed additional data to our deep learning models, which work better with more data than just the 134 images in the initial dataset. If we augment the dataset using the 6 transformations, we now have $6 \times 134 = 804$ samples for training and $6 \times 10 = 60$ for validation – and if in addition, we corner-crop each image and then apply the transformations on each of the crops, we further extend the cardinality of the dataset, with $6 \times 4 \times 134 = 3216$ training samples and $6 \times 4 \times 10 = 240$ validation samples.

B. Baselines

To frame the problem, we first decided to implement baselines, on which we would then base our work to compare performance results.

1) Support Vector Classifier: The first baseline is a non-neural-network approach, using a Support Vector Classifier (SVC) [16], [17], [18], with a RBF kernel. We wanted to see what a more traditional machine learning approach would give on this task – we used `SVC` class from Scikit-Learn to do so.

2) Patch-CNN: The second baseline is a simple 3 layers Convolutional Neural Network (CNN), which directly takes as an input a (16×16) patch and assigns the label $\ell \in \{0, 1\}$ to each one of them. Its architecture is presented in Fig. 3.

3) Vanilla-UNet: This baseline is the one that performed the highest. We chose the U-shaped structure of UNet [4] and implemented it in a similar way to the paper. We used a simple Binary Cross Entropy Loss with it to train.

C. Model Architecture

Transformers, originally mostly used for Natural Language Processing [7], [8], [9], [10], are since 2020 more and more used in computer vision [11], [19]. After analyzing the outstanding results of these applications, we decided to apply a Swin-Transformer [12] to our architecture.

As we used the pre-trained model from the `torchvision` library, we fed the image as input to the Swin-transformer. Before each pooling layer in the Swin architecture (see Fig. 5), we extracted features to allow skip connections. After the last Swin-transformer block, we start deconvolving the image to its original size with the help of the intermediary skip connections previously saved. Then, the decoding step is repeated: up-scaling the feature maps by two while reducing the number of features by 2, concatenating the skip connections, reducing the number of channels by 2, and simply convolving twice with a kernel of size 3.

While the input image goes through the Swin-transformer, in parallel, we convolve twice to obtain 64 feature maps of the same size as the image to create shallow features. We concatenate it and further reduce the number of channels until getting the segmentation map. As the final activation function we use the Sigmoïd function. (see Fig. 6)

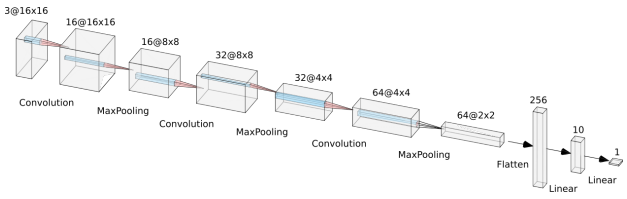

Fig. 3. Patch-CNN architecture – Source: CIL course notebook

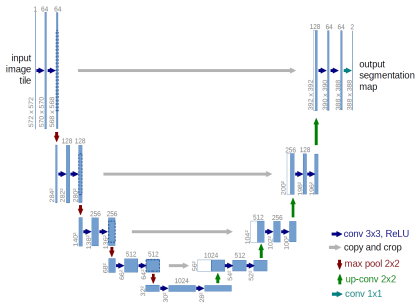

Fig. 4. UNet [4] Architecture

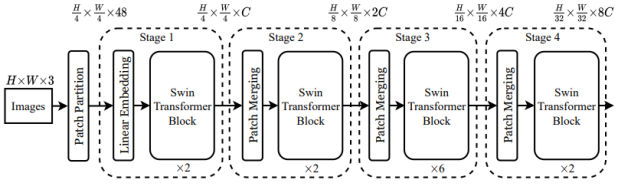

Fig. 5. Swin [12] Transformer Architecture

D. Nontrivial Loss Functions

To train both customized UNet and USwinBaseNet architectures, we decided to leave behind the standard Binary Cross Entropy Loss [20], since, by default, it doesn't address well the class imbalance problematic. Roughly 70% of the dataset masks are not considered as roads (label 0), hence only $\approx 30\%$ of them are roads (label 1), the class we want to segment.

To tackle this issue, we tried different training loops, using the following loss functions – the results are presented in section IV.

1) Smooth Dice Loss: It is defined as follows, with $\text{SmoothSDC}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{2|\hat{\mathbf{y}} * \mathbf{y}| + \gamma}{|\hat{\mathbf{y}}| + |\mathbf{y}| + \gamma}$ being the Smooth Sørensen-Dice Coefficient [21], and γ used for smoothing:

$$\text{SmoothDiceLoss}(\hat{\mathbf{y}}, \mathbf{y}) = 1 - \text{SmoothSDC}(\hat{\mathbf{y}}, \mathbf{y}) \quad (1)$$

2) Mixed Loss: Weighted sum of a binary cross entropy loss (BCE) and a SmoothDiceLoss, with the weights adding up to 1.

3) Mixed Patch F1 Loss: Since the main evaluation metric on Kaggle is the Patch F1 Loss, we built a configurable loss including the Patch F1 Loss (F1Loss_p) and weights (w_1 and w_2):

$$\begin{aligned} &\text{MixedF1Loss}_p(\hat{\mathbf{y}}, \mathbf{y}) \\ &= w_1 * \text{OtherLoss}(\hat{\mathbf{y}}, \mathbf{y}) + w_2 * \text{F1Loss}_p(\hat{\mathbf{y}}, \mathbf{y}) \end{aligned} \quad (2)$$

such that $w_1 + w_2 = 1$

4) Focal Loss: We also tried to train our non-baseline models (UNet and USwinBaseNet) with the Focal Loss (FLoss), which is commonly used to address the issue of class imbalance, using Binary Cross Entropy loss [20]:

$$\text{FLoss}(\hat{\mathbf{y}}, \mathbf{y}) = \alpha(1 - \exp\{\text{BCE}(\hat{\mathbf{y}}, \mathbf{y})\})^{\gamma} \, \text{BCE}(\hat{\mathbf{y}}, \mathbf{y}) \quad (3)$$

5) Focal Tversky Loss: Finally, even though it is also used to address issues of class imbalance, Tversky Loss [22] is more customizable than Dice Loss [23]. It can be defined as :

$$\text{FTverskyLoss} = \left(1 - \frac{\mathbf{TP} + \varepsilon}{\mathbf{TP} + \alpha\mathbf{FP} + \beta\mathbf{FN} + \varepsilon}\right)^{\gamma} \quad (4)$$

It allows to be more granular on the importance we give to each error type ($\mathbf{TP}$, $\mathbf{FP}$, $\mathbf{TN}$, $\mathbf{FN}$), thanks to the weights α, β, ϵ and γ.

The results of training UNet with different losses are presented in section IV

E. Training and Strategy

We trained our models on ETH Zurich's Euler cluster using NVIDIA RTX 2080Ti and NVIDIA GTX 1080Ti, 4-core or 8-core CPU and 16GB to 64GB of RAM. We could select many useful arguments: which loss to use, the number of epochs, save intermediary weights, the batch size, how to (not) augment the dataset, depending on the case and the model we train.

F. Model Evaluation

The model evaluation is done by computing the average Patch-F1-Score on the complete test-set, following the patch division and label assignment described in Data Augmentation subsection. We also collect other metrics, such as training accuracy, validation accuracy, depending on the task we evaluate.

IV. Experiments and Results

A. Results

1) Baselines: Here are the baseline results. SVC performs poorly; without a considerable surprise, Patch-CNN performs a bit better, but its architecture (Fig. 3) is too simple to fit the training data well. All models run on the initial dataset without transformations. Vanilla-UNet runs the best on Kaggle's F1-Score this can be explained by a deeper and more complex architecture, even with the standard parameters. All of them use Binary Cross Entropy Loss [20].

Model	Patch-F1-Score	Train. Accuracy	Val. Accuracy
SVC	0.75080	0.74708	0.76656
Patch-CNN	0.82779	0.84612	0.80026
Vanilla-UNet	**0.88082**	**0.90784**	**0.868498**

TABLE I
METRICS SUMMARY OF OUR BASELINES

2) Our models: We tested the different losses on Vanilla-UNet and USwinBaseNet. As it seemed to perform well with default parameters, we decided to continue with Vanilla-UNet, with changing training parameters. We went through the same process for USwinBaseNet. Note that results obtained from USwinBaseNet were surprisingly unstable, as the different losses can depict it. We kept the best result in the following table, but some were much lower. As we saw a lot of noise in the predictions, we believe that the way we computed the loss didn't allow the network to converge to an optimal solution.

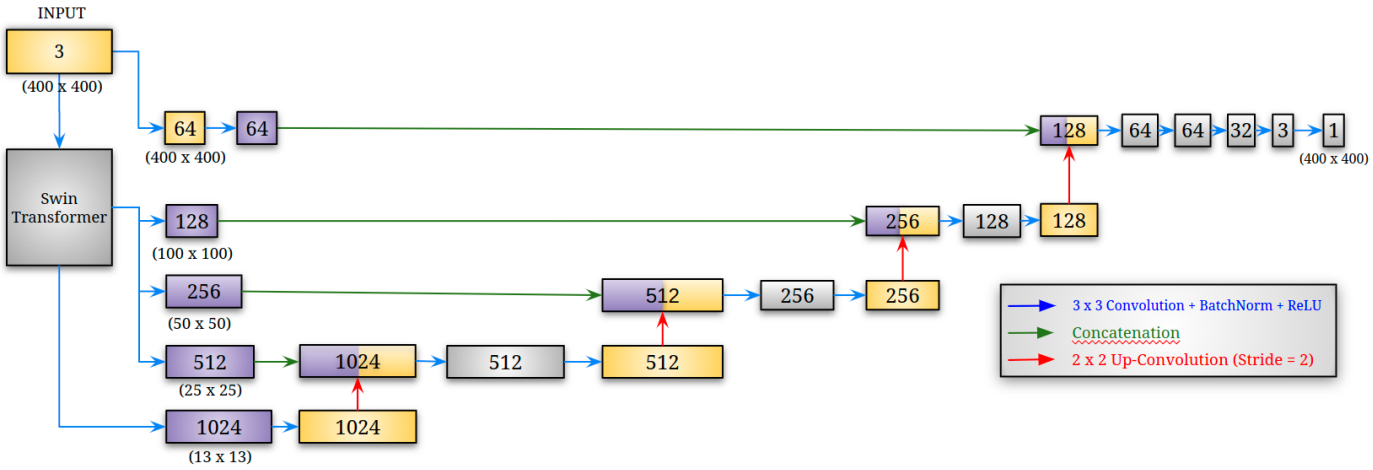

Fig. 6. USwinBaseNet Architecture

Val. Patch-F1-Score w/ Loss	**UNet**	**USwinBaseNet**
BCE (default)	0.88082	0.85373
SmoothDiceLoss	0.89093	0.81520
MixedLoss	**0.91302**	0.86697
FLoss	0.90191	0.86731
MixedF1Loss$_p$ w/ FLoss	0.90602	**0.89502**
FTverskyLoss	0.90877	0.86737

TABLE II
COMPARISON OF LOSSES FOR UNET AND USWINBASENET MODELS

3) Final results: We can see that our fine-tuned version of UNet performs better than our proposed model USwinBaseNet. However, while combining it in ensemble methods, we achieve better results. We believe that our USwinNet architecture predicts too much noise. On the other hand, the UNet predicts clearer and less noisy labels. Our theory is that UNet gives the overall shape of the roads but is completed with more detail from the USwinNet models.

Model	**Best Patch-F1-Score**
SVC (w/ BCE)	0.75080
Patch-CNN (w/ BCE)	0.82779
Vanilla-UNet (w/ BCE)	0.88082
UNet (w/ MixedLoss)	0.91302
USwinBaseNet (w/ MixedF1Loss$_p$)	0.89502
3-ens. of UNet [1]	0.91726
3-ens. of USwinBaseNet [2]	0.90804
4-ens. of USwinBaseNet [3]	0.90191
5-ens. of UNet [4]	0.91771
Ens. of the top 5 subs [5]	**0.91908**

[1] with FTverskyLoss, MixedLoss, FLoss
[2] with 2 × MixedF1Loss$_p$, FLoss
[3] with 2 × MixedF1Loss$_p$, FTverskyLoss, MixedLoss
[4] with all losses above except BCE
[5] with Mix Ens., UNet & USwinBaseNet

TABLE III
BEST PERFORMANCE FOR EACH MODEL

V. DISCUSSION

The proposed model remains unstable, and its performance remains highly variable. We could explain this instability by the fact that we used the pre-trained Swin-transformer. The model was pre-trained on a classification task and trained on ImageNet, a dataset consisting of a thousand classes. However, we believe that this might be a sub-optimal solution to our problem. Instead, we could argue that the model's stability could highly increase by applying more regularization, different loss functions, weight initialization, or, eventually, other pre-trained weights for Swin. Data in this study was mainly composed of aerial images taken from roads located in the same geographic region. Some features of these roads (signalization color, width of the road, environmental backgrounds) might differ from one country to another, leading to lower prediction efficiency when images from remote countries are fed. A way to balance that challenge would have been to get more knowledge about the different features extracted by our architecture. Deep neural networks have long been criticized for being "black boxes", where interpretability of labels is usually not available to a Human Level. We believe that we would have significantly benefited from an explainable model as we would know which areas of the images provoked false predictions [24].

VI. SUMMARY

In this paper, we tried to leverage the power of the Swin Transformer architecture to enhance the accuracy of a deep neural network based on a UNet architecture. The goal was to extract road positions from aerial images. The dataset was composed of 144 images that we augmented to 864 by applying different transformations. One of the first issues we handled was class imbalance, which we solved by using a variety of class imbalance accounting losses. Then we designed a novel architecture using transformers, CNN, and skip connections. Finally, we used ensemble techniques like majority voting to improve our results further. Even if

singularly, our architecture does not outperform the UNet architecture using it in combination with UNet within an ensemble significantly improved our results compared to a single use of one of the networks.

REFERENCES

[1] S. Yuheng and Y. Hao, "Image Segmentation Algorithms Overview," 2017, pp. 1–6.

[2] Z. Zhang, Q. Liu, and Y. Wang, "Road extraction by deep residual u-net," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 5, pp. 749–753, 2018.

[3] E. Maggiori *et al.*, "Convolutional neural networks for large-scale remote-sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 645–657, 2017.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[5] L. Zhou, C. Zhang, and M. Wu, "D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 192–1924.

[6] Y. Wei, Z. Wang, and M. Xu, "Road structure refined cnn for road extraction in aerial image," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 709–713, 2017.

[7] Z. Dai *et al.*, "Transformer-xl: Attentive language models beyond a fixed-length context," in *ACL (1)*, 2019, pp. 2978–2988. [Online]. Available: https://doi.org/10.18653/v1/p19-1285

[8] J. Devlin *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[9] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[10] Z. Yang *et al.*, *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[11] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[12] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," *CoRR*, vol. abs/2103.14030, 2021. [Online]. Available: https://arxiv.org/abs/2103.14030

[13] Z. Zhang *et al.*, "Dcs-transupernet: Road segmentation network based on cswin transformer with dual resolution," *Applied Sciences*, vol. 12, p. 3511, 03 2022.

[14] J. Zhu *et al.*, "Relation-aware siamese region proposal network for visual object tracking," *Multimedia Tools and Applications*, vol. 80, 04 2021.

[15] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach *et al.*, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[16] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, may 2011. [Online]. Available: https://doi.org/10.1145/1961189.1961199

[17] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, 1999, pp. 61–74.

[18] M. Song and D. Civco, "Road extraction using svm and image segmentation," *Photogrammetric Engineering Remote Sensing*, vol. 70, no. 12, pp. 1365–1371, 2004. [Online]. Available: https://www.ingentaconnect.com/content/asprs/pers/2004/00000070/00000012/art00002

[19] Y. Liu *et al.*, "A survey of visual transformers," *CoRR*, vol. abs/2111.06091, 2021. [Online]. Available: https://arxiv.org/abs/2111.06091

[20] U. Ruby and V. Yendapalli, "Binary cross entropy with deep learning technique for image classification," *Int. J. Adv. Trends Comput. Sci. Eng*, vol. 9, no. 10, 2020.

[21] G. A. R. J. C. Carass Aaron, Roy Snehashis, "Evaluating white matter lesion segmentations with refined sørensen-dice analysis," 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-64803-w

[22] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, "Tversky loss function for image segmentation using 3d fully convolutional deep networks," in *Machine Learning in Medical Imaging*, Q. Wang *et al.*, Eds. Cham: Springer International Publishing, 2017, pp. 379–387.

[23] C. H. Sudre *et al.*, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," *CoRR*, vol. abs/1707.03237, 2017. [Online]. Available: http://arxiv.org/abs/1707.03237

[24] Y. Kang, I.-L. Cheng, and W. Mao, "Towards interpretable deep extreme multi-label learning," 2019.