Deep Learning for Avalanche Segmentation

Firas Kanoun, Maxime François Jan

***Abstract*— This project is realized in collaboration with the Computer Vision Lab at EPFL and Lyf startup with the objective of segmenting avalanches from mountain images taken by drones. In this report, we showcase our approach for the segmentation task and present the different results we achieved. We used state-of-the-art deep learning based segmentation models to find the most appropriate architecture for this task. We will show that U-net as a segmentation architecture with Efficientnet or Mobilenet as backbones outperformed other solutions in terms of segmentation quality while being more computationally Efficient.**

I. INTRODUCTION

Lyf startup's objective is to create a device able to autonomously localize buried victims in avalanches using a faster approach than current solutions. With the aim of decreasing intervention time, we are designing an autonomous drone taking on board some principles of avalanche beacons. The device is compact, robust and ergonomic and would improve the process of searching for the buried victim by first taking pictures of the avalanche, then mapping its coordinates to GPS ones and finally scanning it upwards to find the buried victims. This process requires segmenting avalanches from drone images which could be greatly achieved using convolutional neural networks. The task of semantic image segmentation is to classify each pixel in the image. It is different from object detection as it does not predict any bounding boxes around the objects. We do not distinguish between different instances of the same object. For example, there could be multiple mountains in the same scene and all of them would have to have the same label. A higher level understanding of the image is therefore required as the algorithm should figure out not only the objects present but also the pixels which correspond to the object. In this report, we will discuss how to achieve this task by presenting fully convolutional networks and encoder-decoder structures where encoders output tensors containing information about the objects and decoders take this information and produce segmentation maps. We will also dive into the implementation of the pipeline – from preparing the data to building the models and testing them.

II. DATA IMAGES

A. Collecting data

The data we used for this task comes from a dataset of avalanches that we found online [13]. Some of the pictures were taken by humans and contained complete mountain scenes with sky, mountains, trees and skiers as in Fig1 while others were top view pictures taken by drones or helicopters as in Fig2. Although, we planned for our drone to get the highest possible and then take pictures from above, we preferred to train the segmentation model on all images to adapt to any variation that may arise. We first started with a dataset of 70 images and then added 37 images in order to improve the accuracy of the model, for a total of 107 images used. We splitted the images according to the traditional 80/20 for the train and test sets.

Fig. 1: Picture taken by human

Fig. 2: Drone picture

B. Data labeling

As in every supervised machine learning task, models need to be trained to find patterns in data in order to be able to predict outputs for new unseen data. This implies that during the training, models already know about the labels. Since our data consisted of raw images, we started the project by manually going through all of the images and labeling each region. Other approaches based on extracting image patches were tested in this project, we therefore labeled the image as extensively as possible using 9 classes : Sunny/Shadowed avalanche, Sunny/Shadowed snow, Sky, Clouds, Mountains, Trees and Human. Although this labelisation can seem to be relevant when learning by sliding over images and extracting patches, where slight variations in colors can be very meaningful in the learning process, we will see later that it was not in the best profit of learning by CNNs. We therefore later turned to a binary classification labeling only

the avalanches.

C. Data augmentation

Training deep learning neural network models on more data can result in more skillful models, and the augmentation techniques can create variations of the images that can improve the ability of the fit models to generalize what they have learned to new images. We therefore implemented a data generator that randomly applied modification on images at each epoch of the training. These data augmentation techniques were chosen carefully and within the context of our problem. For example, since we don't know the exact rotation that the drone would have when taking a picture, all possible rotation variations were applied. We also took into account the fact that color variations could happen at the moment the drone takes the picture, because of mist or a similar weather condition, by randomly applying blur, saturation and brightness filters to the pictures in each epoch of the training. We also took into account the fact that noise could be added to the picture due to a malfunction for example and randomly added Gaussian noise to all the training pictures. Other data augmentation techniques such as cropping, zooming and translation images were also applied during the training as it can be seen in Fig3. Since neural networks need to have fixed size input images, all images were resized to 480 x 320. We chose this size as it was the most dominant one in our data set in order to avoid as much as possible distorting the images.

(a) Base image (b) Flipped image

(c) Zoomed image (d) Noised image

Fig. 3: Examples of data augmentation

III. ARCHITECTURE

Usually, the architecture of a CNN model contains several convolutional layers, non-linear activations and pooling layers followed by a fully connected neural network. The initial layers learn low-level concepts such as edges and colors and later level layers learn higher level concepts such as different objects. We then map this spatial tensor from the convolution layers to a fixed length vector. However, this destroys all spatial information about pixels, which is not possible for a semantic segmentation task. Taking the low-resolution spatial tensor, which contains high-level information, we have to produce high-resolution segmentation outputs. To do that, instead of mapping the spatial tensor to a fixed size vector, we add more convolution layers coupled with up-sampling layers which increase the size of the spatial tensor. As we increase the resolution, we decrease the number of channels, as we are getting back to the low-level information. This is called an encoder-decoder structure. Usually, the encoder part, also called backbone or base network, is a standard model CNN such as DenseNet, ResNet or EfficientNet. Then, the segmentation architecture is a combination of the encoder and the decoder, which is usually the same network structure as the encoder but in opposite orientation. For this task, 3 different architectures were tested, Segnet, Pyramid Scene Parsing Networks (PSPNet) and Unet. As the ultimate objective of the project is to embed the model in a drone, the idea is to find an architecture that offers the best trade-off between accuracy and complexity. Even though it can be very accurate, a heavy model would not work in our case as this would slow down the inference process when looking for a victim.

A. Models

1) Unet: U-Net takes its name from the architecture, which when visualized, appears similar to the letter U, as shown in Fig4. Altough there is nothing new in the first part of the network, which can be substituted by any backbone as we will see later, the second half of the architecture is quite special. Indeed, the expansion path consists of upsampling of the feature map followed by convolutions that halve the number of feature channels at each step. Then, in order, to keep localization information, cropped feature maps from the encoder are concatenated at each step, which is denoted by the gray horizontal arrows in the architecture.

2) Segnet: SegNet has a similar structure as Unet and consists of symetric encoder and decoder structures. Again the encoder applies convolutions and activation functions on input, but instead of transferring the entire feature maps from encoder to decoder, it only sends the index (location) of the

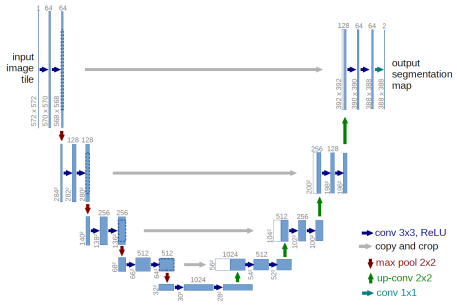


Fig. 4: Unet architecture

value extracted from each window. Then, decoders upsample the input using indices stored from the encoding stage. This reduces drastically the model complexity and thereby the inference time, as there would be no learnable parameters for upsampling, which could be of great help to solve our problem.

3) PSPnet: The Pyramid Scene Parsing Network is optimized to learn better global context representation of a scene. First, the image is passed to the base network to get a feature map. The the feature map is downsampled to different scales. Convolution is applied to the pooled feature maps. After that, all the feature maps are upsampled to a common scale and concatenated together. Finally another convolution layer is used to produce the final segmentation outputs. The main point of PSPnet is that smaller objects are better captured due to the different scales used when downsampling. This can help in detecting very small objects such as trees and skiers in images and exclude them from the segmentation whenever they are present in an avalanche.

B. Loss functions

1) Cross Entropy Loss: We first started the trainings by using the most commonly used loss function in image segmentation, the pixel-wise cross entropy loss. This loss examines each pixel individually, and compares the class predictions to the encoded target masks and is expressed as follows :

$$L(y_i, p_i) = -\sum_{c=1}^{M} y_{i,c} \log\left(p_{i,c}\right) \tag{1}$$

M - number of classes (avalanche, sky, mountain..)

y - binary indicator if class label c is the correct classification for pixel i

p - predicted probability pixel i is of class c

However during the training, we noticed that after a certain number of epochs, the loss kept decreasing but not the validation IOU score anymore. We interpreted this by the fact probabilities were decreasing as the training progressed but that the labeling itself was not changing, which resulted in the observation mentioned above. We therefore included F-score as an additional function in the loss, to put more weight on wrong labels, and then used a weighted combination of both losses. Computational time being very expensive, we could not afford to do a proper cross-validation, but rather chose the combination that gave the best results in the first few epochs.

2) Dice Loss:

$$L(tp, fp, fn) = \frac{2 \cdot tp}{2 \cdot fp + fn + fp} \tag{2}$$

tp - false positive
fp - false positive
fn - false negative

In the multi segmentation task, the F-score is computed as the average of F-scores of each class against the other classes.

C. Metrics

The metric used for evaluating the models throughout this report is the Intersection Over Union (IOU). Simply put, the IOU is the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. This metric ranges from 0–1 with 0 signifying no overlap and 1 signifying perfectly overlapping segmentation.

IV. RESULTS AND INTERPRETATION

A. Multiclassification

The first attempt to tackle the problem was trying to label each region of the image with different classes of a mountain scene, sunny snow, shadowed snow, sky, rocks..etc. However, this attempt was not successful for many reasons. The first one is that many of the classes were under represented compared to other ones, resulting in models that were not able to grasp the features of these regions. For instance, most of the images contained large amounts of sunny snow or skies but very few contained shadowed snow or humans, which resulted in poor IOU scores for these individual classes. In addition, labeling was complicated to do for some mountain images as they contained mixed zones where very small objects co-existed and discerning between them was almost impossible. An example of this is the snow in the mountain of Fig3. Moreover, the classes chosen at the beginning of the project were not objective and open to different human interpretations. For example, we had separate classes for avalanche or snow when in the sun and avalanche or snow when in the shade. This resulted in some arbitration during the labeling process which mislead the model during the learning process, especially that the labeling was shared by two persons. These facts resulted in a poor segmentation as we can see in TABLE I, where well represented classes had good test IOU scores and the under represented ones bad scores. The best result we obtained was 0.53 of mean IOU using Unet structure and Efficientnet backbone. Using a more deep encoder such as efficientnetb4 or b5 would have certainly improved the scores. However, the final objective being to segment avalanches only, we preferred to turn to a binary classification. A summary of the different structures and backbones can be found in TABLE II. More models

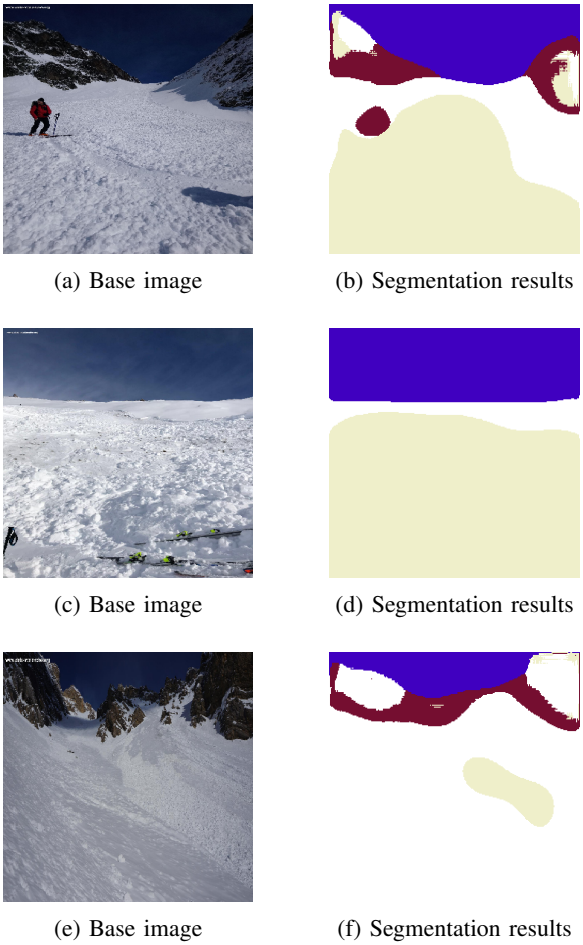
Sunny Avalanche	0.53
Shadowed Avalanche	0.62
Sunny Snow	0.6
Shadowed Snow	0.31
Mountain/Rocks	0.39
Sky	0.72
Clouds	0.76
Vegetation	0.62
Human	0.3
Mean	0.53

TABLE I: Individual IOU scores

could have been tested but due to time constraints and the fact that a training takes more than 6 hours in average, we preferred to concentrate our efforts on the binary case which was more promising. Some of the segmentation results are shown in Fig5. As we can see, well represented classes such as sky are well segmented but humans are not. Rocks were also badly segmented due to the fact that they contained many snow which we believe mislead the models. These results also show that the models were capable of distinguishing the avalanche from the snow, which was quite promising for the next steps.

	ResNet	EfficientNet	DenseNet
Segnet	0.21	0.34	0.19
Unet	0.18	0.53	0.28

TABLE II: Summary of multiclassification results



(a) Base image (b) Segmentation results

(c) Base image (d) Segmentation results

(e) Base image (f) Segmentation results

Fig. 5: Some results of the multiclassification

B. *Binary classification*

Based on the multi classification results, it was likely that Unet and Efficientnet were probably the best combination for our problem. We therefore concentrated our efforts on these two structures trying Efficientnet backbones of different depths. Another structure was also introduced in the binary case, PSPNet. This was in the purpose of better recognizing smaller objects such as human or trees and eliminating them from the segmentation zone of the avalanche. Aiming at reducing model complexity for the reasons mentioned earlier, we also tried the Mobilenet backbone. This model was proposed by Google and is optimized for having a small model size and thereby a faster inference time. This is ideal to run on mobile phones or resource-constrained devices such as microcomputers in our case.

Resorting to binary classification improved by much the IOU score due to the fact that there were no more arbitrage of sun and light as it was the case for the multi classification. The different combinations tried are showcased in Fig8.
The dots representing backbones of Unet structures and the crosses backbones of PSPnet structures, we can again infer the superiority of Unet in solving our problem compared to other models. These results also confirmed that Efficientnet as a backbone was the best option in terms of IOU score. However, this may not be the best option considering the complexity-accuracy trade off. Indeed, Mobilenet, denoted by a red dot in the plot, was able to reach a comparable IOU score of 0.84 with approximately 7 times less parameters. Considering that with a 8GB-memory computer, it takes approximately 20 sec to load the weights and 3 seconds to infer a segmentation, efficientnetb5 may not be the best option. In addition, when embedding the model in a Rasberry Pi, one should expect the inference time to be multiplied by 3 to 4 times.

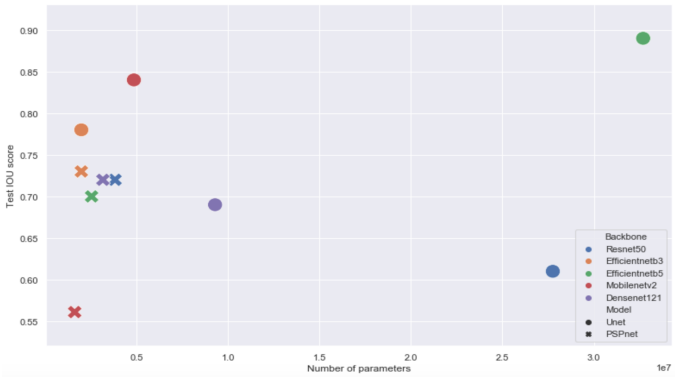
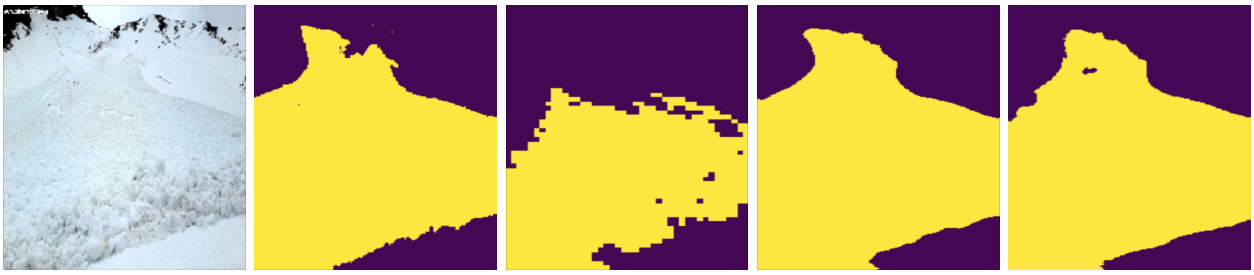


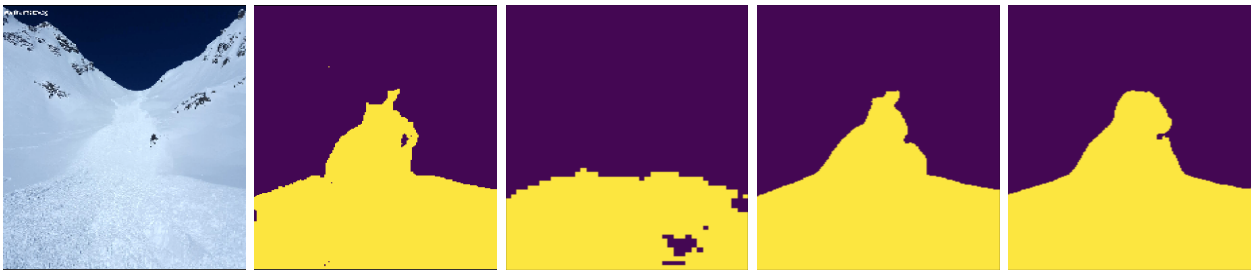
Fig. 8: Results of binary classification

The segmentation results of the main models are showcased in the images of Fig6 and Fig7. These inferences are results on test images. In accordance with the scatter plot of Fig8, Unet can almost perfectly segment the avalanche, especially with Efficientnet or Mobilenet as backbones. On the other hand, PSPnet was not always successful as we can see in the figures. Aiming at improving even more the accuracy of the inference, we added a denseCRF to the pipeline in order to smooth the segmentation and remove the isolated small parts from the segmentation. This filter operated by comparing the segmentation mask to the initial image and then removing the noisy pixels around the edges in order to smooth the segmentation results. The computation is very fast and does not add any latency burden to the



(a) Base Image (b) True Mask (c) Psp densenet121 (d) Unet efficientnetb5 (e) Unet mobilenetv2

Fig. 6: Examples of results of binary classification



(a) Base Image (b) True Mask (c) Psp densenet121 (d) Unet efficientnetb5 (e) Unet mobilenetv2

Fig. 7: Examples of results of binary classification

inference. The objective of the drone being to localize the avalanche, we wanted to have a single biggest connected component of pixels, which we succeeded to have thanks to the denseCRF. However, this did not improve by much the IOU score due to the fact that CNNs were already very precise and that the filter sometimes removed parts that were relevant to the segmentation. As a matter of fact, it only improved the IOU score by 1% on average.

V. CONCLUSIONS

In this project, we explored the most recent models and techniques in image processing and computer vision in order to solve a real world problem. We used state-of-the-art deep learning based segmentation models in order to infer an avalanche mask given any mountain picture. The objective being to embed the model in a micro computer, we focused on low complexity models that were capable to segment the avalanche without being too computationally expensive. In this context, we believe that there is still room for some improvements which can be achieved through quantization techniques. This would half the memory needed to make an inference and thereby reduce the waiting time to rescue a victim. At the time of writing this report, we are currently focusing our efforts on embedding the Mobilenet model into a raspberry Pi and make sure that inference can be achieved in a reasonable amount of time.

REFERENCES

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation” .

[2] Philipp Krähenbühl, Vladlen Koltun, ”Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials”.

[3] Badrinarayanan, V., Kendall, A., Cipolla, R. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation"

[4] A. Garcia-Garcia, S. Orts-Escolano, S.O. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, ”A Review on Deep Learning Techniques Applied to Semantic Segmentation”.

[5] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, Jiaya Jia, ”Pyramid Scene Parsing Network”.

[6] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, ”MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”.

[7] Mingxing Tan, Quoc V. Le, ”EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”.

[8] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, ”Densely Connected Convolutional Networks” .

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, ”Deep Residual Learning for Image Recognition”

[10] github.com/cvlab-epfl/LabelGrab

[11] github.com/qubvel/segmentation-models

[12] github.com/luyanger1799/Amazing-Semantic-Segmentation

[13] www.data-avalanche.org/