Machine Learning Project 2

Rémy Hidra - 119034990010

I. Introduction

In Computer Vision tasks, deep convolutionnal neural networks are becoming increasingly useful. They provide a flexible framework to execute different tasks. In the last few years, CNNs performed state-of-the-art scores in common image classifications. Open sourcing theses pre-trained deep neural networks helped the scientific machine learning computer vision community build coherent experiments with easy to use models. Fine tuning and transfer learning with these complex models helps generalize learned image features to many various tasks, including image generation, detection and segmentation.

In this work, we present various modifications and experiments to analyse the performances of common state-of-the-art machine learning models to solve multiple computer vision problems. VGG is a very popular CNN. It is small and simple to understand and implement. Therefore, it can be easily modified for other tasks and does not need a high computing power to fine tune. ResNet is a more recent CNN, which realized state-of-the-art performances on the ImageNet classification data set. It uses a complex residual convolution block. This makes it harder to use, but the network can understand more abstract image features, which makes it ideal to reach a better accuracy.

In section II, we use a ResNet and a VGG models to classify a CIFAR10 data set. In section III, we explore the image generation field using a VAE on two image data sets. In section IV and V, we solve an image segmentation problem on the PASCAL VOC data set, using, respectively, a VGG and a ResNet modified network.

II. Image classification

In this section, Jiaqi Yu solves an image classification problem. To compare two different architectures, we implemented a VGG16 and a ResNet networks. ResNet is made of residual blocks as shown in figure 1. VGG16 is a classic CNN, made of 16 main blocks. The basic architecture of the network is shown in figure 2. The image classification task is done on the CIFAR10 image data set.

The experiments are done with a fixed epochs number, SGD momentum optimizer and a varying learning rate.

From the results, we can see that the ResNet model is performing better than the VGG. This is expected as the ResNet model is a lot more complex and deep than VGG. This also confirms the state-of-the-art observations. With ResNet, we obtain a testing accuracy of $83.73\,\%$ with a learning rate of 0.1 and an accuracy of $87.62\,\%$ with a learning rate of 0.05. With VGG, we obtain a testing accuracy of $80\,\%$ with a learning rate of 0.1 and an acuray of $84.6\,\%$ with a learning rate of 0.05. We can see that the performance is better with a smaller learning rate. This is certainly caused by the high number of epochs.

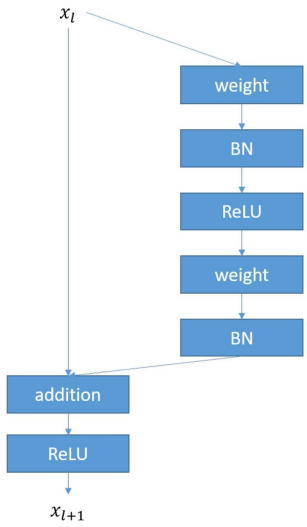


Fig. 1: Example of a residual convolution block from the ResNet architecture

Additionally, we perform a t-SNE and PCA visualization of the 10 vector output of both models, using test data. This allows for a better understanding of the inner working of the model. We see that the clustering done through t-SNE is a lot better than the one done by the PCA. Less aliasing is also visible.

III. Image generation

In this section, Benshun Yin is implementing an image generation algorithm. We are particularly interested in the Variationnal Auto Encoder (VAE) structure. It is made of an encoder and a decoder. The encoder convert the input image in a lower dimension vector. The smaller encoded space is called the latent space. Then, a decoder is increasing the encoded sample size to try to reconstruct the original image. The main structure is shown in figure 4. To experiment with the model, we used two data sets, MNIST and CIFAR10.

First, we experiment different loss function. We try a cross entropy (CE) loss, a mean squared loss (MSE) and a sum of both loss (CE + MSE). From the results, we can see that the CE + MSE case attains the smallest loss, followed closely by the MSE case.

Second, we tried to vary the latent space size. With bigger dimensions, as expected, we can decrease the loss of the model. The resolution of images obtained is also higher.

Fig. 2: VGG16 architecture used for image classification of CIFAR 10

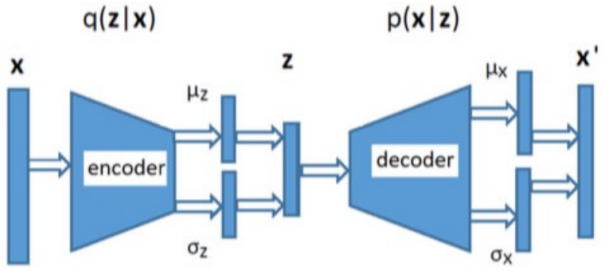


Fig. 3: Structure of the VAE network used in section III

Third, we use different learning rate. We can see that the learning of the model is faster when the learning rate is higher. We find an optimal learning rate at 0.0005.

Finally, we operate the feature visualization algorithms PCA and t-SNE on the input data and encoder output. We find that the distance between each cluster is unchanged during the encoding process.

IV. Image segmentation with VGG

In this section, Claire Laverne is doing image segmentation using a VGG16 backbone. The encoder is based of a VGG16 network and the decoder is made symmetrically, using deconvolutions layers. Instead of pooling layers, the model uses unpooling layer, a special structure which tries to preserve lost information in the encoder. Figure **??** shows the architecture of the model. The data set used to train the model is the image segmentation data set of the PASCAL VOC set.

We first experiment different optimizers for the image segmentation task. Between a SGD, SGD with momentum and an Adam optimizer, we compared the loss, accuracy curves and some samples results. We found that the best accuracy, less noisy curves and most satisfying segmentation results are obtained by using an Adam optimizer.

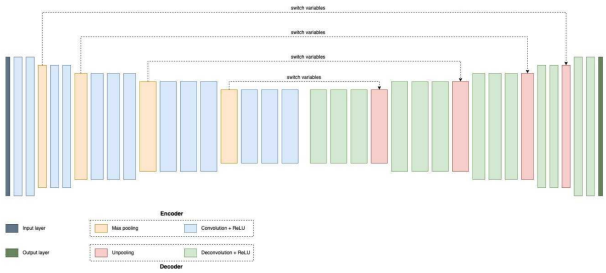


Fig. 4: Architecture of the image segmentation model based on VGG16 used in section IV

To better understand how the decoder works, we visualize few feature maps at the end of each decoder block. We notice that the unpooling layer helps to enlarge the image, while the deconvolution allow to increase the resolution of the image.

We apply a t-SNE transformation to better visualize the data distribution after multiple stages in the model. We can see that the model helps to reform clusters of data, which where lost during the encoding. The obtained distribution is similar to the expected distribution, which shows the model is working properly.

Finally, we examine the confusion matrix of the test data. We can see some weaknesses in our model, with some categories that are never predicted. Some confusions between classes also happen.

V. Image Segmentation with ResNet

In this section, we will focus in more depth in the work of Rémy Hidra, which was to solve an image segmentation problem, using a ResNet backbone image segmentation encoder decoder model.

A. Model

As said before, a standard image segmentation neural network is made of two part. The encoder, or backbone, is extracting features in the image into a low-resolution matrix with a high depth. The decoder is increasing the resolution of the feature vector, to obtain a segmentation mask. In this section, we will use a smaller version of ResNet50 as a backbone and a modified variation of U-net as the decoder. Those two networks present interesting features.

The ResNet model is made of multiple *residual* blocks. More precisely, it is made of *residual identity layers* and *residual convolutionnal layers*. The former conserves the size of the input volume unchanged, while the latter reduces the size of the volume and increases its depth. It is a standard network for image classification and it is also used as a feature extractor for many computer vision tasks.

The U-net model is known in the image segmentation field, as one of the networks to implement a concatenation in the decoding layers. This strategy is used in order to re-sample information lost in the encoding process, back in the decoder. We will implement only the decoding part of the U-net model and the concatenation connections.

Our model is composed of an encoder made of a pre-trained ResNet50 network on the ImageNet database. The ResNet has been truncated by half in order to accelerate the training process. We then build a decoder inspired by the U-net model, with up-sampling layers, followed by concatenations of up-sampled and down-sampled layers.

We used the Pascal VOC data set for image segmentation. It features 2 913 training samples in the 2012 VOC and 210 annotated testing samples in the 2007 VOC. The segmentation set has 20 different classes. Our network classify each pixel as one of theses or as background. The input is a $(256, 256)$ RGB image, while the output is a $(256, 256, 21)$ matrix of probabilities of each pixel belonging to one of the 20 classes or the background. We show a representation of our model in figure 5.

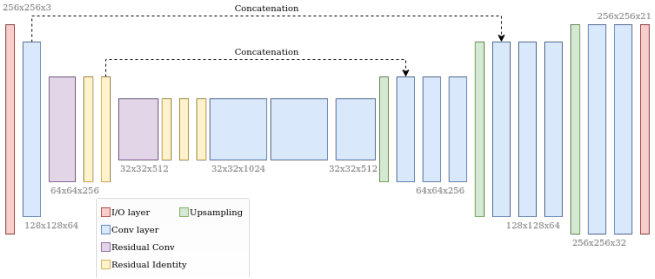


Fig. 5: Representation of our model, a combination of ResNet and U-net

We did four different experiments on our model. In section V-B, we compare different optimization techniques, then section V-C analyze the imbalanced data distribution. In section V-D, we compare how the concatenations inspired by U-net improves our model. Finally, in section V-E, we interpret results from some layers of our network using a t-SNE visualization.

B. Optimizer comparison

In this section, we try to compare different gradient descent techniques. We will experiment with a stochastic gradient descent (SGD) with a learning rate of 0.0001 (**A**), a SGD with a learning rate of 0.001 (**B**) and an Adam optimizer with a 0.001 learning rate (**C**). Both A and B optimizer use a $\gamma = 0.9$ momentum. All the models are using the 0.1 background class weight explained in section V-C and are trained during 582 epochs with 5 mini-batch of 32 samples per epoch. We present the results of this experiment in figure 6 and table I.

For reasons of data imbalance discussed in section V-C, the accuracy is not an ideal measure of the quality of learning. Instead, it is useful to compute the *precision*. It is computed as the ratio $P = \frac{\#\text{true positives}}{\#\text{total positive}}$. Therefore, it is robust to data imbalance, because it only looks at the correct classifications.

	Optimizer	Learning rate	Loss	Accuracy	Precision
A	SGD	0.0001	0.732	62.36 %	**96.21 %**
B	SGD	0.001	**0.487**	**71.52 %**	92.87 %
C	Adam	0.001	0.679	59.97 %	93.10 %

TABLE I: Numerical results when varying the optimizer between models A, B and C, as described in section V-B

By looking at the figure 6a, we can see, as expected, that the high learning rate of the model B helps to reach a lower local minimum than the model A. The model C converges the fastest, while the A is quite slow. A stable precision, in figure 6c, is reached almost instantly by the model B and C, which are fast to converge thanks to a high learning rate. The model A is much slower. It only reaches stability after 220 epochs. However, in the end, the model A performs a higher precision than the other two. Contrarily, the model B reaches high scores in accuracy and loss. But since the data distribution suffers from a strong data imbalance, those results are not strictly representative of the real performances of the model. Therefore, until the end of this section, we will use a SGD optimizer with a 0.0001 learning rate to optimize our model training.

C. Imbalanced data analysis

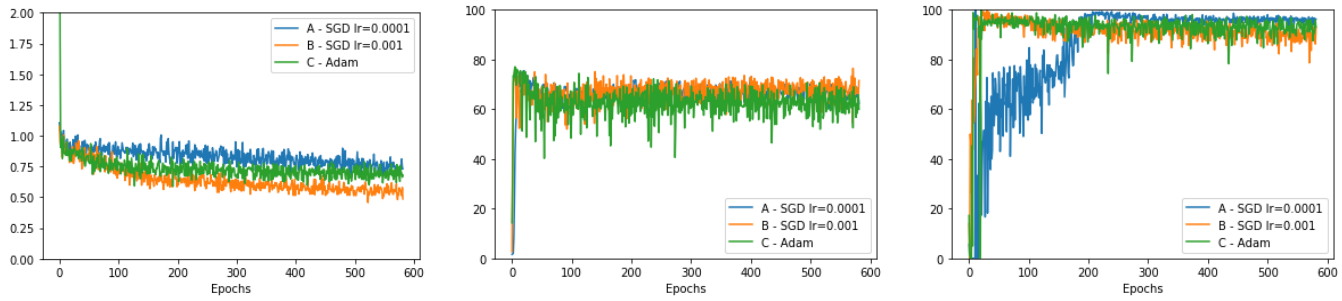
Image segmentation classification data sets often suffer from a major problem of data imbalance. Indeed, in the Pascal VOC data set, the majority of pixels are labeled as background. Therefore, without treating this issue our model has a risk of learning a wrong solution to our problem. For example, if our class distribution is made of $90\,\%$ of background classes, the model can obtain a $90\,\%$ accuracy by just classifying every pixel as background. A new loss function has to be built, in order to penalize the model correctly. A classic machine learning solution to fix data imbalance is to apply weights in the loss function. In this section, we try to visualize the consequences of this class distribution imbalance and we experiment different ways to solve it.

We build four different models. They all use a categorical cross entropy loss function, with optional weights multiplying each classes coefficient. The models uses respectively (**A**) no weights, a 0.1 weight on the background class (**B**), a 0.01 weight on the background class (**C**) and weights proportional to the quantity of each class in the training data set (**D**).

After training we obtain a precision of 0.841 for model B, 0.7752 for C and 0.842 for D. We have the biggest precision on model B and D.

In figure 7, we can see the outputs of the four models on two unknown samples from the testing data set. The model A performs as expected. Because no weight has been applied on the loss function, the model labels every pixel as background. The white pixels are labeled as *person* class, which is also appearing a lot in the training data set. Thus, it is coherent to see pixel labeled as this class when no weights are applied.

With the models B and C, we see the other classes appearing as expected. Even if the pixels are mislabelled, the shape of the expected mask is more recognizable. We can assume that adding more layers in the network improves greatly the segmentation. A lot more classes appear with



(a) Loss functions **(b)** Accuracy functions **(c)** Precision functions

Fig. 6: Loss, accuracy and precision curves for models A, B and C along 582 epochs

the model C. The background may have lost too much importance in the loss function to classify pixels correctly.

In the model D, the normalization of each class made the classification process very complicated for the model. A lot of classes appear, while the background and person classes are not as predominant as before.

In figure 8, we can see the confusion matrix of the models. Similar observations can be made here. The model A tries to classify most pixels as background or person. With models B and C a better equalization can be seen. Interestingly, the bright horizontal line at the top of the figure 8c shows that the weight applied to the background in the model C may be too high, because it misclassified a lot of background pixels. Finally, with model D we obtain the most pixels in the diagonal line, which gives us the most precision. However, the background class weight is still too strong, which makes the model unable to properly form the shape of each mask.

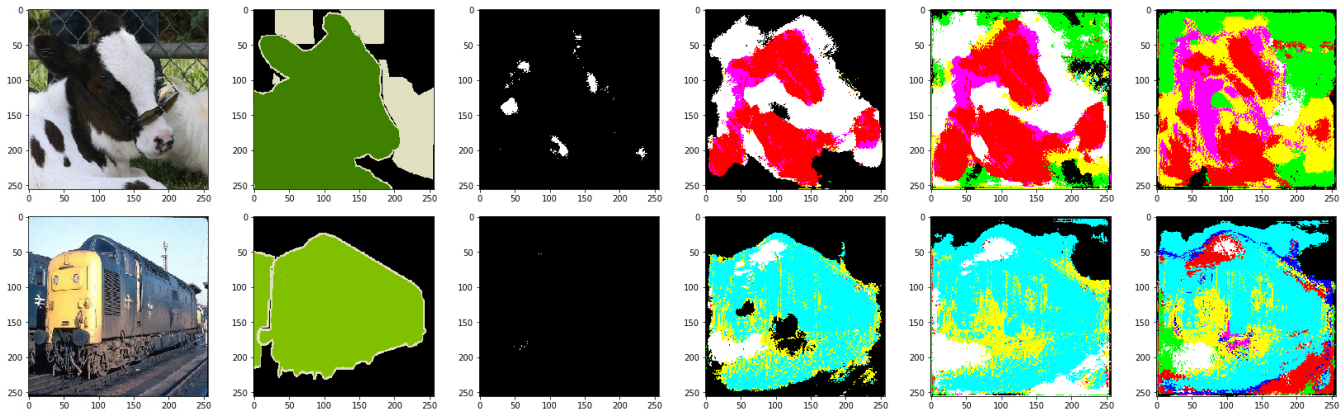
To conclude, our model is indeed not powerful enough to understand the training data and classify correctly each pixel without weights applied to the loss function. Guessing the correct distribution of class weights is a difficult task. We saw that, while a distribution dependant on the proportion of each class may be satisfying to maximize the classifying precision, it degrades the shape detection of the model.

D. Influence of data concatenation

When building our model, we were heavily influenced by the U-net model. This model introduces a structure with connections between the encoder and the decoder. Indeed, features volumes from the encoder are concatenated to the output of certain layers of the decoder. This characteristic of the model is supposed to help remind the decoder of certain features of the original image, which may have disappeared along the encoding process. It worth noting that this connection between the encoder and decoder can be widely seen in modern state-of-the-art image segmentation models.

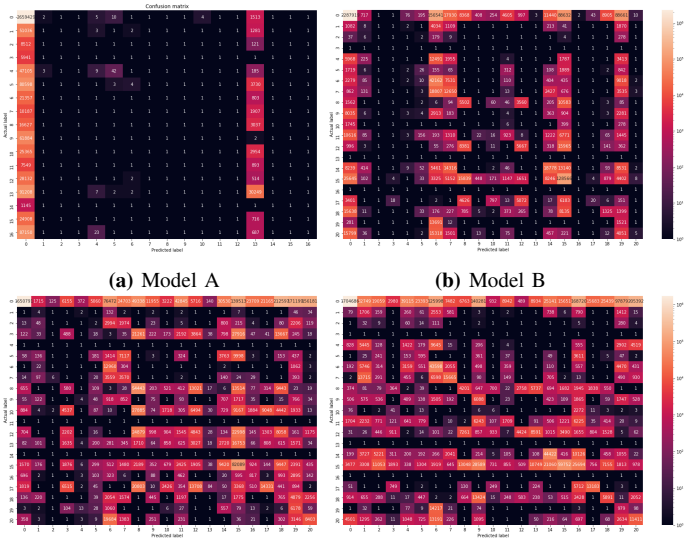
In this section, we try to observe the influence of this feature in our model structure. Therefore, we compare a reference model built before (**A**), with a modified version of the model which does not have concatenations operations (**B**).

In figure 9, we see a comparison of the model A and B outputs on two different testing samples. With the first sample, the mask of the boat is quite small, which makes it hard to recognize for the model. Moreover, during the encoding process, the multiple down-samplings through convolutions makes it possible for small pixel information to be lost in forward pass. As expected, we see that the boat shape is detected by the model A, while it was lost



(a) Input image **(b)** Expected mask **(c)** Model A **(d)** Model B **(e)** Model C **(f)** Model D

Fig. 7: Results on two testing samples of the class weighting experiment described in section V-C



(a) Model A **(b)** Model B

(c) Model C **(d)** Model D

Fig. 8: Confusion matrix for the models A, B, C and D. The height maps are displayed on a logarithmic scale. The x-axis is the predicted class and the y-axis is the expected class for each pixel.

in the model B. Therefore, the concatenation connections are a useful part of the network to remind possible lost information.

Another interesting fact can be seen on the second sample in figure 9. We can see that the output of the model A is a bit noisy, especially on the edges of the mask. Contrarily, the model B has smoother edges. This effect can be seen on most testing samples. This may be due to the concatenation connections adding noise to the network, which may remind one of classic noise inducing layers, like the dropout layer. Thus, concatenation may be a simple way to also improve the generalization skills of the model.

E. Feature t-SNE visualization

Machine learning tasks usually feature a lot of high dimensional data. Thus, it is useful to use tools to reduce the number of dimensions and better understand relationships between data samples. Similarly, while we can easily interpret the inputs and outputs of our image segmentation model, it is hard to understand the hidden convolutionnal layers. In this section, we try to visualize internal features extracted by the network. This can be interesting in order to understand the interaction between the encoder and the decoder, especially our ResNet encoder which is pre-trained on the ImageNet classification data set.

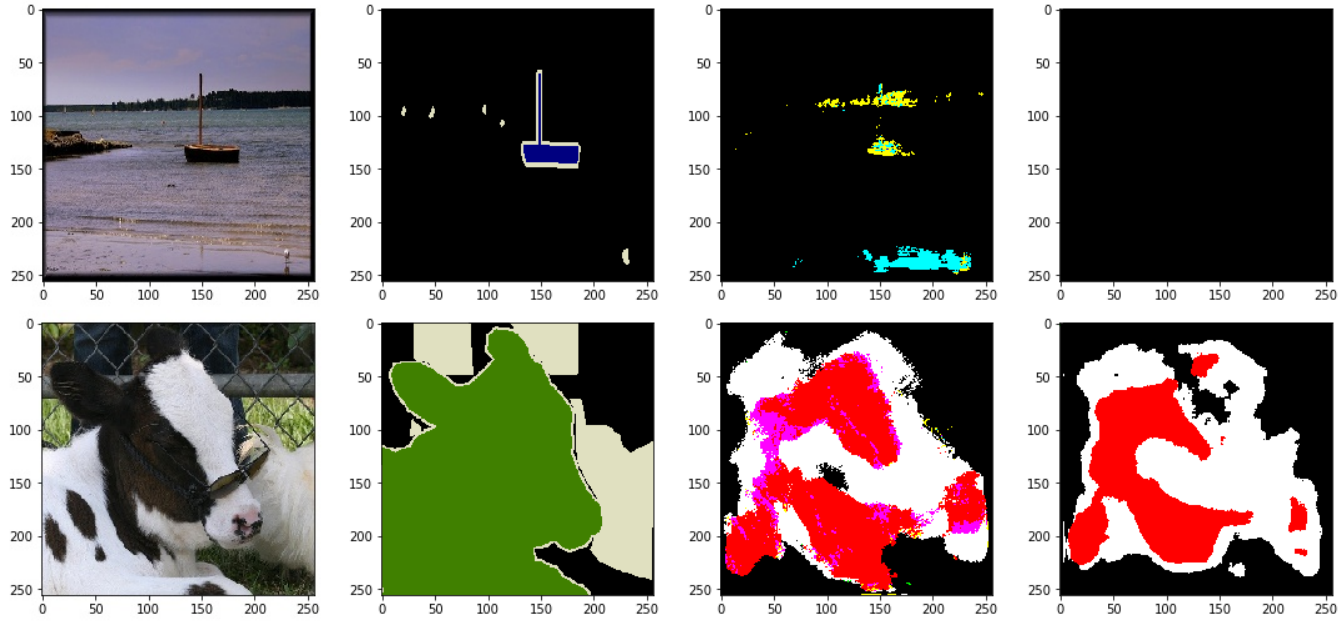
We use a reference model and we extract intermediate features layers of multiple data points. Then, we use a t-SNE algorithm to reduce the dimensionality of the data to a 2D plane. First, we look at the features extracted by the last layer of the encoder, in figure 10.

The t-SNE visualization of the encoder output is not very satisfying. If we isolate some classes, like with figure 10b, we can see some clusters. However, for most classes, most samples are too spread out across the visualization to represent anything. Therefore, it is hard to interpret anything of the understanding of our data by the ResNet encoder. On the one hand, this can be due to the truncation of the model. By removing some top layers, we may have removed critical components to extract valuable information in the images. On the other hand, the t-SNE visualization may not work properly with the features we extracted.

In figure 11, we visualize an intermediate layer in the decoder layer, in order to interpret the learned behavior of the model. Here, the visualization is more satisfying. In figure 11b, we see the same cluster as in figure 10b. This shows that the decoder did not lose the valuable clustering done by the encoder. Similarly, by comparing figures 10c and 11c, we can see that the decoder helped create a new cluster for the class *bus*. Therefore, we can assume that the decoder understood the information stored in the encoder features. The t-SNE visualization may not be ideal in this case.

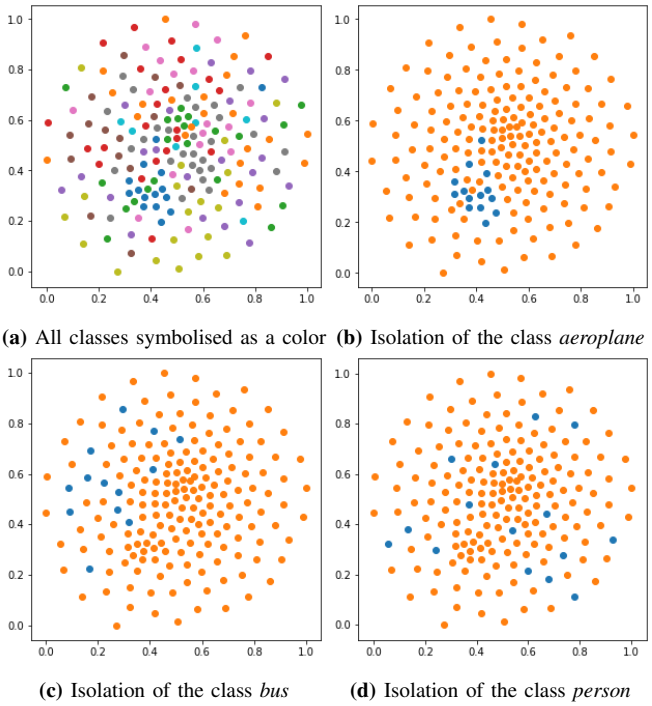
VI. Conclusion

In our work, we have seen good performances in image classification, reconstruction and segmentation. However, a lot of different structures have not been examined. The GoogleNet network is very popular in image classification, while GANs and more recently transformers model performs well as image generation models. Lastly, plenty of improvements have been seen in segmentation, with pyramidal structure models.



(a) Input image **(b)** Expected mask **(c)** Model A **(d)** Model B

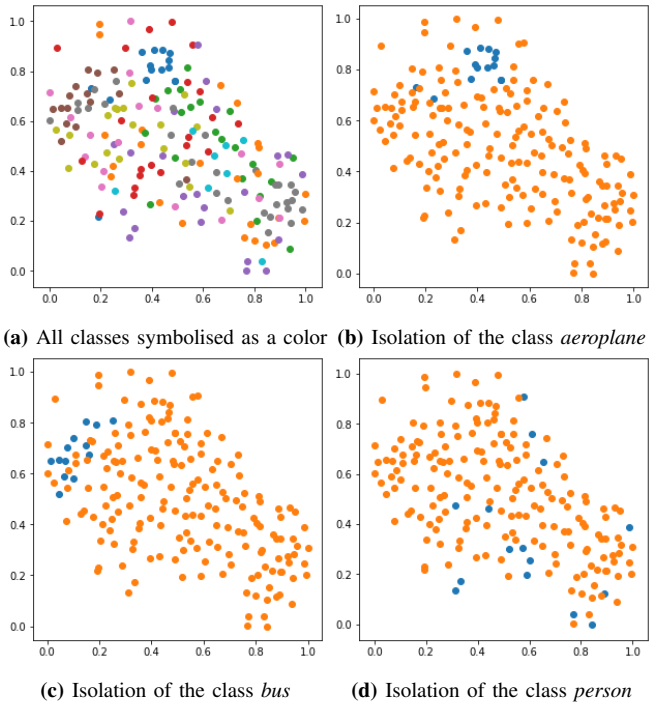
Fig. 9: Comparison of the results with and without the concatenation links in the model structure described in section V-D



(a) All classes symbolised as a color **(b)** Isolation of the class *aeroplane*

(c) Isolation of the class *bus* **(d)** Isolation of the class *person*

Fig. 10: t-SNE visualization in a two dimensional plane for the feature layer at the end of the encoder for 192 training samples



(a) All classes symbolised as a color **(b)** Isolation of the class *aeroplane*

(c) Isolation of the class *bus* **(d)** Isolation of the class *person*

Fig. 11: t-SNE visualization in a two dimensional plane for the feature layer in the middle of the decoder for 192 training samples