Project Report

Luigina Ferrara, Anna Gargiulo, Ida Maruotto

***Abstract*—Upon adapting the U-Net network to perform joint intensity classification and specimen segmentation on HEp-2 cells, the network has been further modified by changing its backbone with the ResNet34 network, and adopting the Gradient Normalization algorithm for training. This paper evaluates the performances obtained on the HEP2 dataset using a combined loss - one for each task.**

I. Introduction

In recent years, deep machine learning approaches have become popular in the medical field to help discriminate between sick tissues and healthy tissues. These approaches can be used for epithelial type 2 cells, and their objective is to perform segmentation and classification of these cells. A famous network developed for image segmentation purposes is U-net, composed of an encoder part and a decoder part. The version proposed in this paper is slightly changed, allowing to perform the image segmentation and classification at the same time, exploiting common weights coming from the encoder part of the network. This network is defined as Joint U-net [3]. The dataset considered is the HEp-2 Images Dataset, composed of Indirect ImmunoFluorescence (IIF) stained hepitelial cells for the diagnosis of Autoimmune Diseases. Indirect ImmunoFluorescence is a test for antinuclear antibodies (ANA) analysis. IIF slides are examined at the fluorescence microscope, and their diagnosis requires both the estimation of fluorescence intensity and the description of staining pattern. For the purpose of this project, the network should, at the same time, produce a segmented version of the images, classify the label of the images and their fluorenscence intensity values. For these individual tasks,

- the segmentation of the images is produced after the images have been analyzed by both the encoder and decoder part of the U-net;
- the label classification should assign each image to a specific class, corresponding to ANA staining patterns, chosen amongst a set of 7:
  1. homogeneous: diffuse staining of the interphase nuclei and staining of the chromatin of mitotic cells;
  2. speckled: granular nuclear staining of interphase cell nuclei;
  3. nucleolar: large coarse speckled staining within the nucleus, less than six in number per cell;
  4. centromere: several discrete speckles (40-60) distributed throughout the interphase nuclei and characteristically found in the condensed nuclear chromatin.
  5. golgi: discontinuous speckled or granular perinuclear ribbon-like staining with polar distribution in the cytoplasm [1];
  6. numem (Nuclear Membrane) homogeneous staining of the nucleus with greater intensity at its outer rim;[1]
  7. mitsp (Mitotic Spindle) the spindle fibers between the poles are stained in mitotic cells [1].

  Example of each class is shown in Fig 1.
- the intensity classification should, instead, determine if the fluorescence has an intermediate (weak) or positive (strong) level. The intensity classification can also have a negative level, but the dataset does not have samples of this specific class since no other analysis can be carried out in presence of such samples.

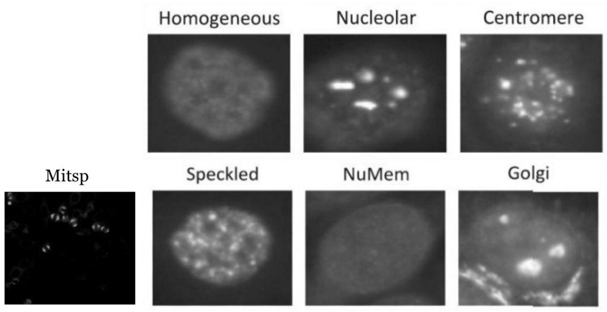

Fig. 1. Reference image for each class

II. Description of the Problem

Multi-Task learning (MTL) is a deep learning approach which allows to solve multiple tasks at the same time while exploiting commonalities across the tasks (shared representation), even if the task type is different. By using MTL, performance improvement for the related tasks can be achieved. Plus, the generalizability of the system is obtained because learning multiple tasks forces the model to focus on the features that are useful across all of the tasks and more information on the problem is available. The main goal of the experiment is to exploit the advantages of the Multi-Task learning through the Joint U-net, a MT network which perform both the image segmentation and classification. The main focus of the paper is to describe the performances of the joint U-net on the HEP2-cells dataset when the architecture of the network is further changed. U-net takes its name from its shape, so by keeping the encoder-decoder ratio, this network can still be used for medical approaches. The encoder of the network has to be substituted with the reference module of the ResNet34, a network trained for image classification. The decoder should be symmetrical, so it is necessary to evaluate the second part of the network so that it matches the first. Since the problem is multi task (segmentation and classification of images), the approach to be used to compute the weights of the loss of the network is the Gradient Normalization Algorithm, designed to find the best weights when a network has to solve more than one problem. This algorithm has a unique settable parameter, α. The estimation of the performance needs to be evaluated with two different sets of losses. Both of them are complex losses, and only differ for the loss of the segmentation task. For the first set, the loss chosen is the dice loss, based on Sørensen-Dice score, and for the second set the binary cross entropy (BCE) is used for the segmentation.

III. Methods

To fulfill the requirements of the problem, it is first necessary to define the loss for the project. The loss exploited is defined as a MultiTask Loss, since every single task is described by a specific loss. To be thorough, the task is taken upon with two different multi task losses.
For the first approach, the complete loss is defined as so:

- binary cross entropy, for the intensity classification of the images;
- cross entropy, for the label classification of the images;
- dice loss, for the segmentation of the images.

For the second approach, the loss is, instead, defined as:

- binary cross entropy, for the intensity classification of the images;
- cross entropy, for the label classification of the images;
- binary cross entropy, for the segmentation of the images.

The loss is defined as a class that instantiates the binary cross entropy, the cross entropy and the dice losses. The *forward* function returns a unique tensor, equal to the concatenation of the single tensors of the single losses.

Moreover, to train the network the Gradient Normalization algorithm [2] was necessary. The algorithm automatically balances training in deep multitask models by dynamically tuning gradient magnitudes. It also matches or surpasses the performance of exhaustive grid search methods, despite only involving a single asymmetry hyperparameter α.

The constant α sets the strength of the restoring force which pulls tasks back to a common training rate. In cases where tasks are very different in their complexity, leading to dramatically different learning dynamics between tasks, a higher value of α should be used to enforce stronger training rate balancing. When tasks are more symmetric, a lower value of α is appropriate [2]. In this paper, three different α values have been tested: 0.06, 0.6 and 6 as it, in theory, can be any positive value.
The GradNorm can be used to train the whole network by following the steps described in Fig 2. The approach used was to define a Gradient Normalization class where the training for each epoch is handled.

Algorithm 1 Training with GradNorm

```
Initialize $w_i(0) = 1 \ \forall i$
Initialize network weights $\mathcal{W}$
Pick value for $\alpha > 0$ and pick the weights $W$ (usually the
    final layer of weights which are shared between tasks)
for $t = 0$ to $max\_train\_steps$ do
    Input batch $x_i$ to compute $L_i(t) \ \forall i$ and
        $L(t) = \sum_i w_i(t) L_i(t)$ [standard forward pass]
    Compute $G_W^{(i)}(t)$ and $r_i(t) \ \forall i$
    Compute $\overline{G}_W(t)$ by averaging the $G_W^{(i)}(t)$
    Compute $L_{\text{grad}} = \sum_i |G_W^{(i)}(t) - \overline{G}_W(t) \times [r_i(t)]^\alpha|_1$
    Compute GradNorm gradients $\nabla_{w_i} L_{\text{grad}}$, keeping
        targets $\overline{G}_W(t) \times [r_i(t)]^\alpha$ constant
    Compute standard gradients $\nabla_{\mathcal{W}} L(t)$
    Update $w_i(t) \mapsto w_i(t+1)$ using $\nabla_{w_i} L_{\text{grad}}$
    Update $\mathcal{W}(t) \mapsto \mathcal{W}(t+1)$ using $\nabla_{\mathcal{W}} L(t)$ [standard
        backward pass]
    Renormalize $w_i(t+1)$ so that $\sum_i w_i(t+1) = T$
end for
```

Fig. 2. Training with Gradient Normalization

Before performing any training, the requirements for the project specified that the backbone of the network had to be changed. In Fig 3 the classic joint U net model is shown.

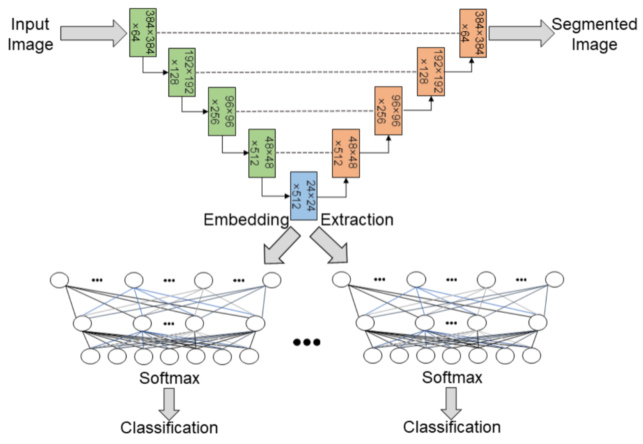

Fig. 3. Original Joint U-Net Architecture

The encoder of the network (shown in green and blue in Fig. 3) is made of several *Down* blocks (shown in Fig. 4, each constituted of a MaxPooling layer, and a Double Convolutional Layer (a double sequence of Conv2D, Batch Normalization and ReLU layers). The decoder part (shown in orange in Fig. 3) is responsible of reconstructing the mask from the embedding representation (the output of the blue block in Fig. 3). The embedding layer also represents the starting point from which other two network parts classify labels and intensities. These three tasks share the encoder (the backbone).

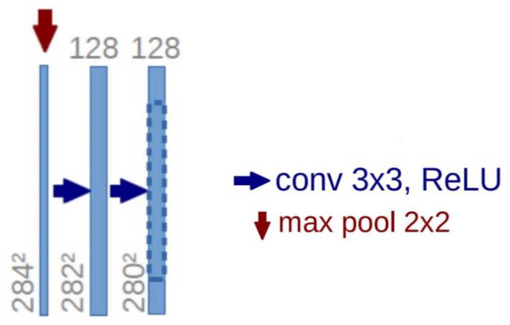

Fig. 4. Example of the Down block

To introduce the ResNet34 in the joint U-net model, it was first necessary to identify the module block of the network.

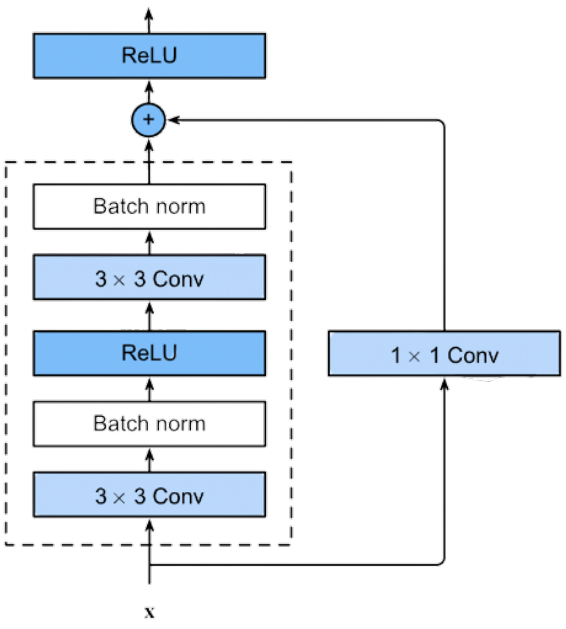

Fig. 5. ResNet34 Module Block

The block is characterized by a similar layout with respect to the original *Down* Block, with the addition of a skip connection that is linked to the output through a Convolutional layer of Kernel 1x1, that is needed to adapt the dimensions of the output to the original dimensions of the input.
These layers have been encapsulated into a *BasicBlock* that takes the place of the *Double Convolution Layer* in the *Down* block of the network. The choice is motivated by a desire of keeping the module of the ResNet separated from the logic of the U-Net encoder.
Moreover, it is necessary to keep a downsampling layer (like the MaxPooling layer), so that the encoder of the network still performs a downsampling of the images before feeding the information gathered to the upsampling and classification parts of the network.

After defining the basic structures to perform the training, the HEp-2 dataset has been split into folds so that it would be possible to estimate the performances of the modified net using Cross Validation.
The dataset is composed of 15120 images, with each image belonging to a different patient. There are 252 patients in total. For each patient there are 4 specimens, segmented into 15 patches, the atomic units of the dataset. The selected Cross Validation is composed of 5 folds. Since it is not possible to divide patches that belong to the same specimen or divide specimens that belong to the same patient and insert some into the training set and others into the test set, all the images of the same patient need to be put in the same set. To do so, the division has been performed on the patients rather than on the patches. After selecting the patients that should belong to a fold, all their images are moved into the correct set.
The patients are selected using a pseudo-random algorithm with seed 12357. This operation has produced 5 sets, split into training and test sets for each fold, as shown in Table I.

TABLE I
TRAINING AND TEST SETS IN EACH FOLD

Folds	Train	Test
A	2,3,4,5	1
B	1,3,4,5	2
C	1,2,4,5	3
D	1,2,3,5	4
E	1,2,3,4	5

The dataset is imbalanced with respect to the classes of label classification with this division:

- 3180 homogeneous;
- 3060 centromere;
- 3120 speckled;
- 3000 nucleolar;
- 1260 numem;
- 900 mistsp;
- 600 golgi.

The dataset is imbalanced with respect to the classes of intensity classification too, with this division:

- 8520 intermediate;
- 6600 positive.

IV. Training and Results

Having defined the methods, another step is necessary before starting the actual training: the definition of the hyperparameters. Some of those parameters remained the same in different tests, while other changed. The stable hyperparameters are:

- In the training phase, 80% of the samples are used to train, and the other 20% is used to validate the training;
- The number of image channels specified is 3;
- The optimizer used is Adam, with an initial learning rate of 0.0001;
- The images are fed to the network in batches of size 8;
- The dimensions of the images is defined (width = 384; height = 384);
- A threshold, used to define whether the network has segmented and classified the image correctly or not, is set to 0.5;

- For the sake of simplicity, the number of output classes for the segmentation task is set to 1, as it can be seen as a binary classification problem (background: values lower than the threshold, foreground: values higher than the threshold).

Parameters that changed include:

- The number of epochs;
- Segmentation Loss (Binary Cross Entropy or Dice);
- α, the only hyperparameter of the GradNorm algorithm.

The eight trainings performed will be described divided into 4 categories. Each of them is composed of two networks identical to each other except for the loss of the segmentation task.

A. I Training

The first experiment was carried out with the following parameters: 30 epochs and α = 0.6.

This approach was used for both the Binary Cross Entropy MultiTask loss and the Dice MultiTask loss. The results of this training are shown in Table II and in Table III.

TABLE II
DICE ACCURACY FOR EACH FOLD USING ALPHA 0.6 AND 30 EPOCHS

Accuracy	FoldA	FoldB	FoldC	FoldD	FoldE
Mask	0.967	0.965	0.972	0.972	0.973
Label	0.798	0.793	0.808	0.799	0.847
Intensity	0.821	0.851	0.751	0.810	0.915

TABLE III
BCE ACCURACY FOR EACH FOLD USING ALPHA 0.6 AND 30 EPOCHS

Accuracy	FoldA	FoldB	FoldC	FoldD	FoldE
Mask	0.960	0.965	0.976	0.974	0.972
Label	0.815	0.806	0.804	0.809	0.859
Intensity	0.852	0.841	0.757	0.816	0.925

At the end of the training, it was possible to have a look at the accuracy plots on each fold. An example is shown in Fig 6, where the accuracy on Fold A using the Dice Multi Task loss is shown. Other folds' plots are not reported as they have similar trends and main characteristics with respect to fold A's.

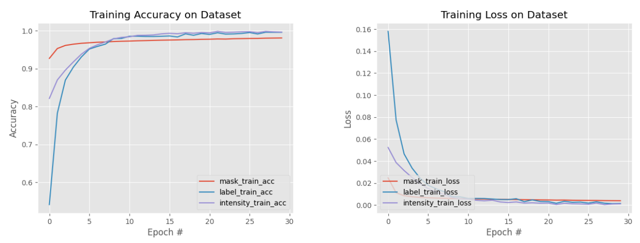

Fig. 6. (Left) accuracy on Fold A, (right) loss on Fold A, Dice Multi Task Loss, $\alpha = 0.6$.

From the plot of the accuracies and the losses, a certain stabilization of the values is evident. Moreover, the accuracy of intensity and label classification are over 0.99 in such plateau, whereas the losses are below 0.05 starting from the 13^{th} epoch. Since these trends might be a sign of overfitting, it was decided to re-train these two networks with a lower number of epochs. The plot of the losses and accuracies of the other folds, not reported here, have identical trends as fold A.

This analysis on the network trained with Dice Multi Task loss is also valid on the network trained with BCE Multi Task loss, as it shows the same behavior for both losses and accuracies.

B. II Training

The second training was executed with the same α = 0.6, but the number of epochs was reduced to 15. The results of this training are shown in Table IV and in Table V.

TABLE IV
DICE ACCURACY FOR EACH FOLD USING ALPHA 0.6 AND 15 EPOCHS

Accuracy	FoldA	FoldB	FoldC	FoldD	FoldE
Mask	0.960	0.955	0.973	0.970	0.972
Label	0.816	0.745	0.783	0.818	0.847
Intensity	0.851	0.806	0.761	0.812	0.847

TABLE V
BCE ACCURACY FOR EACH FOLD USING ALPHA 0.6 AND 15 EPOCHS

Accuracy	FoldA	FoldB	FoldC	FoldD	FoldE
Mask	0.966	0.964	0.972	0.971	0.972
Label	0.800	0.806	0.783	0.809	0.834
Intensity	0.832	0.861	0.751	0.834	0.902

The results are quite the same as in the I training, they deviate with a maximum of 0.01. Since the training is significantly faster having half of the number of epochs, it was decided to keep training the networks with different values for $alpha$, but with the fixed number of epochs equal to fifteen.

Losses' and accuracies' plots are not reported as they have ordinary trends. As expected, they are identical to the I training, but cut off at epoch 15^{th}.

C. III Training

In this case, the number of epochs is fixed to 15, but the value chosen for $alpha$ is 0.06. A trial with this value for $alpha$ was considered since the tasks are medically very related, so these results can tell if the classification tasks can have huge benefit from segmentation task, or viceversa. The results of this training are shown in Table VI and in Table VII.

TABLE VI
DICE ACCURACY FOR EACH FOLD USING ALPHA 0.06 AND 15 EPOCHS

Accuracy	FoldA	FoldB	FoldC	FoldD	FoldE
Mask	0.965	0.964	0.970	0.966	0.970
Label	0.8	0.84	0.784	0.810	0.82
Intensity	0.858	0.830	0.758	0.811	0.897

TABLE VII
BCE ACCURACY FOR EACH FOLD USING ALPHA 0.06 AND 15 EPOCHS

Accuracy	FoldA	FoldB	FoldC	FoldD	FoldE
Mask	0.963	0.963	0.969	0.971	0.973
Label	0.816	0.823	0.794	0.802	0.854
Intensity	0.751	0.865	0.762	0.808	0.918

With this training, a relevant improvement or downgrading of the performances of the networks was expected, but it was possible to conclude that the change from 0.6 to 0.06 is not significant, since the results are quite the same.

Losses' and accuracies' plots are not reported as they have ordinary trends.

D. IV Training

Taking into consideration the results of the III training, the number of epochs remained 15, but α was set to 6, a change in the value of the parameter that should be noticeable at the end of the training. This $alpha$ value, in broad terms, ought to mean that the tasks are very different from each other and can be treated individually. This means that they can be let diverging from each other thanks to a high $alpha$ exponent, that magnifies their differences. The results should downgrade, since the tasks are, in reality, correlated to one another.

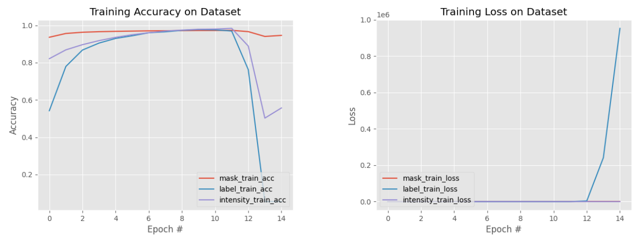

Fig. 7. (Left) accuracy on Fold A, (right) loss on Fold A, Dice Multi Task Loss, $\alpha = 6$.

From the plot of the accuracies (Fig 7), it can be noticed that the accuracy starts falling from the 11^{th} epoch. The loss, consequently, increases a lot from 13^{th} epoch, changing the scale of the plot, making smaller values indistinguishable from each other in the plot. They are reported in Tab VIII.
It is important to mark that loss values are valid and relevant even though they are elevated. In fact, only the intensity loss, as it is evaluated with categorical cross entropy, can assume values higher than one, though both dice loss and binary cross entropy can only be positive values lower than one.

TABLE VIII
DICE LOSS FOR FOLD A USING ALPHA 6 AND 15 EPOCHS

Epoch	Dice Loss	Label loss	Intensity loss
1	0.217	1.26	0.414
2	0.11	0.621	0.310
3	0.094	0.370	0.25
4	0.085	0.268	0.202
5	0.080	0.201	0.162
6	0.761	0.157	0.126
7	0.072	0.120	0.101
8	0.071	0.102	0.086
9	0.068	0.084	0.067
10	0.067	0.069	0.054
11	0.066	0.075	0.050
12	0.064	0.094	0.043
13	0.085	3476	0.21
14	0.155	241367	0.693
15	0.138	952752	0.6926

This behavior might be due to overshooting caused by α value, in this case, bigger than one.
The results of testing this network are shown in Table IX and in Table X.

TABLE IX
DICE ACCURACY FOR EACH FOLD USING ALPHA 6 AND 15 EPOCHS

Accuracy	FoldA	FoldB	FoldC	FoldD	FoldE
Mask	0.931	0.953	0.875	0.958	0.954
Label	0.078	0.04	0.176	0.26	0.06
Intensity	0.588	0.479	0.694	0.435	0.62

TABLE X
BCE ACCURACY FOR EACH FOLD USING ALPHA 6 AND 15 EPOCHS

Accuracy	FoldA	FoldB	FoldC	FoldD	FoldE
Mask	0.940	0.949	0.973	0.938	0.959
Label	0.156	0.04	0.819	0.16	0.22
Intensity	0.600	0.479	0.771	0.637	0.62

As we expected, this last training did not yield good results on the test set.

V. CONCLUSION

The overall scores of the networks are reported in Table XI and in Table XII, where a summary of the mean accuracies for the mask segmentation task, the intensity and label classification for each network are shown.

TABLE XI
MEAN ACCURACIES FOR EACH MODEL USING ALPHA 0.6 VARYING NUMBER OF EPOCHS

Model	Mask accuracy	Label accuracy	Intensity accuracy
Dice (30)	0.970	0.809	0.823
Dice (15)	0.966	0.801	0.815
BCE (30)	0.969	0.818	0.832
BCE (15)	0.969	0.806	0.836

Since accuracy is not the best measure to check the performance of a network trained with imbalanced dataset in the medical field, it was chosen to plot the Receiver Operating

TABLE XII
MEAN ACCURACY FOR EACH FOLD ON 15 EPOCH VARYING ALPHA

Model	Mask accuracy	Label accuracy	Intensity accuracy
Dice (0.06)	0.967	0.811	0.831
BCE (0.06)	0.967	0.817	0.820
Dice (6)	0.934	0.122	0.563
BCE (6)	0.952	0.279	0.621

Characteristic (ROC) Curve of the specific networks. They are shown with the addition of the calculus of the Area Under The Curve (AuROC), which can be used as a performance value for the intensity classification task. The conclusion of this analysis is shown in Fig 8.

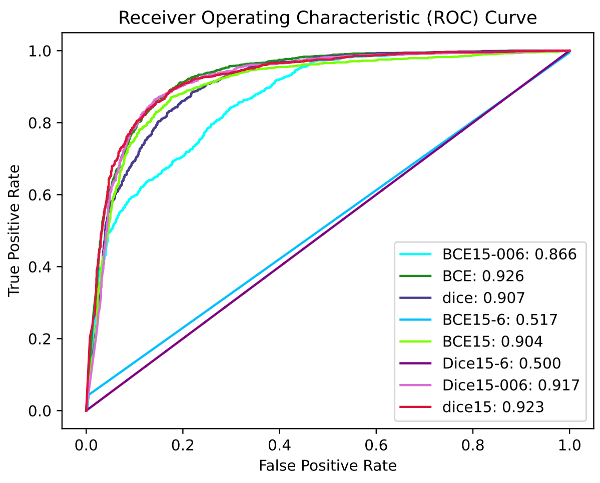

Fig. 8. ROC Curve of the 8 networks and their AuROC on Fold E. The name of the labels correspond to: "Name of the segmentation loss" "Number of epochs of the train" - $alpha$ value without the dot. The labels "BCE" and "dice" refer to the networks of the I training, with number of epochs equal to 30. In networks with label without $alpha$ value specification, the constant is set to 0.6.

From plot in Fig 8, the best resulting network for intensity and label classification is the network trained with 30 epochs and $alpha$ set to 0.6, using the Binary Cross Entropy MultiTask loss. For mask segmentation, the best performing network is the one trained using the Dice MultiTask loss with 30 epochs and $alpha$ set to 0.6. In this regard, it is important to highlight that the execution outperforms the network that is the best for classification task only by 0.001.

Conversely, the worst performing networks are the ones characterized by the Dice and Binary Cross Entropy losses, trained with 15 epochs and $alpha$ equal to 6, that, as the plot for intensity shows, behave as random classifiers. Thus, they are of no use.

Looking at networks trained with 15 epochs, the best performing on segmentation and intensity classification task is the one trained with BCE Multi Task Loss and $alpha$ equals to 0.6, whereas on label classification the best performing it is the one trained with BCE Multi Task Loss and $alpha$ equals to 0.06.

The conclusion is that the tasks are somewhat related to one another and values of $alpha$ lower than 1 can benefit the two classification tasks.

REFERENCES

[1] Information derived from *ICAP: International Consensus on ANA Patterns.* https://www.anapatterns.org/

[2] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, Andrew Rabinovich. *GradNorm*: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks https://arxiv.org/pdf/1711.02257.pdf

[3] Gennaro Percannella, Umberto Petruzzello,Pierluigi Ritrovato,Leonardo Rundo,Francesco Tortorella, Mario Vento *IEEE: Joint Intensity Classification and Specimen Segmentation on HEp-2 Images: a Deep Learning Approach* https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9956212&tag=1