Report - Final Study Project

Pierre Minier
pminier@bordeaux-inp.fr

eVida Laboratory
University of Deusto - Bilbao, ES

1st February 2023 — 30th June 2023

Binary Classification of Neuroblastoma using Convolutional Neural Networks

Abstract

This report presents a study of tumour classification on histological images using Convolutional Neural Networks (CNN). The application framework is neuroblastoma, a malignant tumour developing from nerve cells and mainly affects infants and young children. Chapter 1 mentions already existing algorithms for neuroblastoma classification and provides an overview of CNN technology. Chapter 2 proposes a method for partitioning a database with a limited number of images of varying dimensions to create independent subsets. Chapters 3 and 4 explore the optimisation of CNNs, via fine-tuning techniques, to achieve tumour classification. The results suggest that modified architectures of Inception3 provide the best performance on binary classification, with scores of around 88% and a loss around 0.3. Chapter 5 describes how the simulations were carried out and supervised, and presents a carbon footprint.

Keywords: classification, CNN, fine-tuning, neuroblastoma, partitioning databases

Ce rapport présente une étude de classification de tumeurs sur des images histologiques en utilisant des réseaux de neurones à convolution (CNN). Le cadre applicatif est le neuroblastome, une tumeur maligne se développant à partir de cellules nerveuses et affectant principalement les nourrissons et les jeunes enfants. Le chapitre 1 mentionne des algorithmes déjà existants permettant la classification du neuroblastome, et fournit un apercu de la technologie des CNN. Le chapitre 2 propose une méthode pour partitionner une base de données avec un nombre limité d'images de dimensions variables afin de créer des sous-ensembles indépendants et diversifiés. Les chapitres 3 et 4 explorent l'optimisation des CNN, via des techniques de fine-tuning, pour parvenir à classifier des tumeurs. Les résultats montrent que des architectures modifiées de Inception3 donnent les meilleures performances sur de la classification binaire, avec des scores de l'ordre de 88% et une perte autour de 0.3. Le chapitre 5 explique de quelle manière les simulations ont été réalisées, et présente un bilan carbone.

Mots clefs: classification, CNN, fine-tuning, neuroblastome, partition de données

Contents

List of Terms and Acronyms

Glossary

batch size Number of training samples used during one optimization.

epoch Iteration over all the train dataset for optimising the model.

ImageNet Large-scale database of labeled images containing over 14 million images.

learning rate Step size at which the model adjusts its parameters during the optimization.

momentum Parameter for smoothing out gradients, by adding a fraction of the previous one.

patch Squared cut from an image with higher dimensions.

sample Image from a ready-to-use dataset.

Acronyms

CNN Convolutional Neural Network.

D Differentiated.

IFP Iterative Filling Partitioning.
IFPO Iterative Filling Partitioning with Overfill.

PD Poorly Differentiated.
PSP Proportional and Sequential Partitioning.

ResNet Residual Network.

SGD Stochastic Gradient Descent.

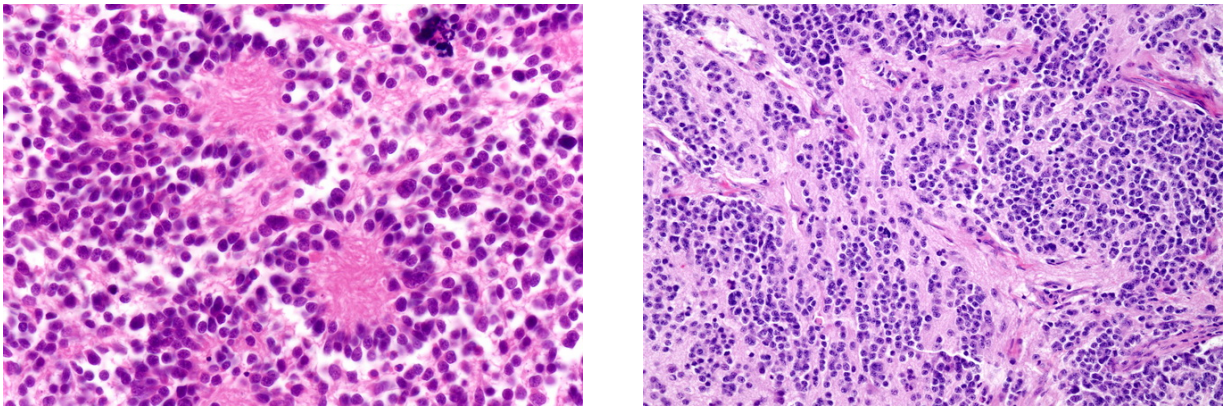
U Undifferentiated.

VGG Visual Geometry Group.

WSI Whole Slide Image.

Introduction

Neuroblastoma is a type of cancer developing from immature cells of the sympathetic nervous system, and is often diagnosed in children under 5 years old [1]. This cancer is relatively rare, but can be aggressive and spread rapidly to other parts of the body by metastasis. Diagnosis is usually made by histological imaging (Figure 1), which provides an overview of the composition of the tumour cells, including whether they are mature or immature. The so-called immature cells are more likely to spread to other parts of the body, and therefore make the disease more severe.



(a) immature cells

(b) less immature cells

Figure 1 – Histological images (cells are in purple)

To assist medical doctors in grading the severity of neuroblastoma, a classification system called Shimada was developed in the late 90s [2]. This system is based on the aspect of the tumour cells observed, taking into account their morphology and arrangement. The cells are then classified according to several categories, ranging from those that are mature (less aggressive), to those that are immature (more aggressive). Nevertheless, histological analysis is a complex and tedious task, requiring both expertise and time. Therefore, these two objectives drive this project:

1. Speeding up the diagnostic process for a more effective management of children with this disease.
2. Reducing human errors by enhancing the accuracy of classification.

The investigated solution involves the use of machine learning algorithms, and especially Convolutional Neural Network (CNN). CNNs are algorithms capable of processing large quantities of images and recognising complex features in the data. These algorithms are therefore particularly well suited to histological analysis, as they can automatically identify tumour cells and classify them according to the Shimada system. The patterns extracted by CNNs are not necessarily the same as the ones used by pathologists. It is therefore possible to improve the accuracy and robustness of diagnoses, in contrast to more traditional approaches that rely on algorithms mimicking the doctor's behaviour.

Algorithms at the state of the art in the classification of neuroblastoma on histological images are discussed in Chapter 1 in order to provide some points of comparison. A database is provided to perform the learning of CNN, but it suffers from a number of defects and irregularities. Solutions are developed in Chapter 2, and can be extended to other databases with similar characteristics: few images of varying size, with high dimensionality. Finally, the search for the best CNN model is presented in Chapters 3 and 4.

Chapter 1: State of the Art

1.1 Neuroblastoma Tumors

1.1.1 The Shimada System

The Shimada system is a histopathological method used to diagnose and categorize neuroblastoma, a cancer that impacts the sympathetic nervous system. It was developed by Dr. Shimada, and is based on the evaluation of histopathological images of the tumour. The Shimada system [2] is widely used and may be considered as one of the most accurate and reliable methods for classifying neuroblastoma.

Figure 1.1 provides a simplified overview of the system, that categorises tumours into two groups. In the case of neuroblastoma, the final classification is based on two markers. One is the aim of this work: the Grade of Differentiation of the tumour cells (Undifferentiated, Poorly Differentiated, or Differentiated).

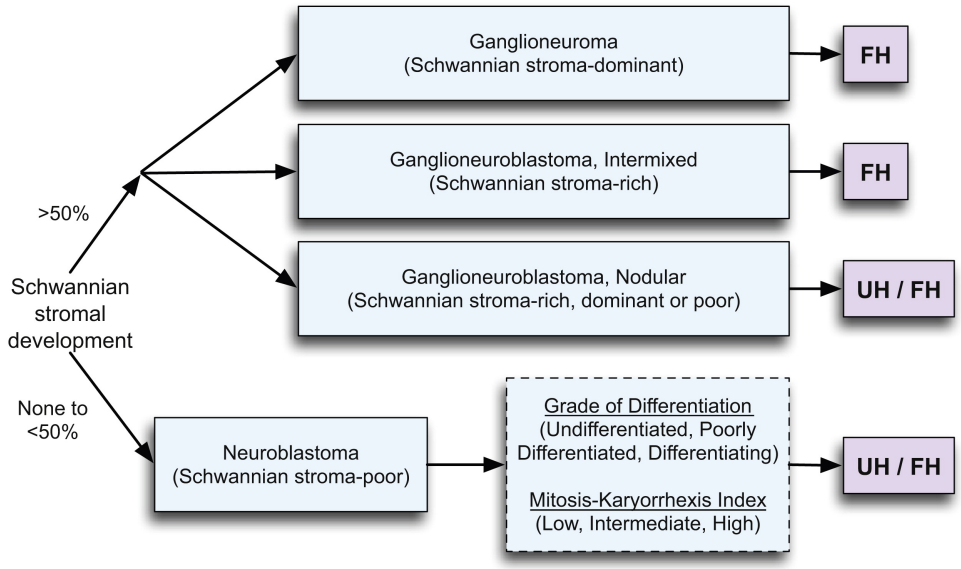


Figure 1.1 – The Shimada classification.
FH: Favourable Histology, UH: Unfavourable Histology

According to [2], a higher level of differentiation is associated with a more favourable prognosis. In fact, tumour cells with poor differentiation are associated with a lower survival rate due to their less mature nature, which makes them challenging to treat. These cells are characterised by a higher rate of growth and a greater tendency to spread to other parts of the body, reducing the chances of a successful outcome. In addition, poorly differentiated cells exhibit increased resistance to chemotherapy and radiation therapy, further complicating efforts to achieve a cure.

1.1.2 Whole Slide Images

Histology is the study of tissues and cells at the microscopic level. It is a fundamental discipline in anatomy and biology that involves the examination of tissues to understand their structure and function. Whole Slide Images (WSIs) are digital versions of these tissues. They are created by scanning stained tissue sections at high resolution, and capturing an image of the entire slide that can be numerically analysed. Generally, the classification of histopathological images is a challenging task due to variations in illumination and shape.

In [3], various general methods for feature extraction and classification applied to WSIs were discussed, providing insight into the underlying concepts and commonly used algorithms in the preprocessing stage.

With regards to colour features, common techniques include the use of colour histograms, colour moments (mean and standard deviation), and colour coherence vectors. Different colour spaces are often tested to

improve performances. Texture features, on the other hand, are extracted using common methods such as CM (Co-occurrence Matrix), LBP (Local Binary Patterns), and Gabor filtering. The combination of CM and Gabor filters is also a popular method, while LBP is frequently used due to its rotation and grey invariant characteristics. Other texture features used include correlation, entropy, average, and variance. In the case of deep learning features, CNN are commonly deployed for feature extraction, followed by the use of classifiers such as SVM (Support Vector Machines).

1.2 Existing Algorithms

This section delves into the examination of four specific methods for the classification of the Grade of Differentiation based on Neuroblastoma WSIs.

1.2.1 Classifying Grade of Neuroblastic (October 2008)

The article [4] presents a computer-aided evaluation of neuroblastoma on WSIs with the objective of classifying the grade of neuroblastic differentiation. The dataset consisted of 36 cases of neuroblastoma, covering all three subtypes of neuroblastic grading defined by the Shimada System (fig 1.1).

The images were processed at various scales, starting at a resolution of 64×64 and increasing as necessary, until 512×512. This mimics the method used by pathologists and saves computation time.

Semantic segmentation in the Lab color space was used to divide the image into five classes: nuclei, cytoplasm, neuropil, red blood cells, and background. The segmentation employed a clustering-based approach, maximizing the interclass distance over the intraclass distance.

Only two regions were retained: cytoplasm and neuropil. Based on it, 24 features were constructed to describe the current scale-level studied, utilizing four kind of metrics (entropy, mean, variance, and homogeneity) across the three channels (L, a, b). To improve accuracy, the features from the current scale-level and lower scale-levels were combined into a pool, chosen through the Sequential Floating Forward Selection (SFFS) algorithm.

Finally, a voting and weighting procedure combined the decisions made by a collection of classifiers. The highest accuracy obtained was 88%.

1.2.2 Patched Completed Local Binary Pattern (April 2018)

The Patched Completed Local Binary Pattern (PCLBP) method [5] presents another approach for classifying histopathological images. The study uses a dataset of 1043 images from The Tumour Bank at Kids Research at The Children's Hospital at Westmead.

The proposed method consists of four steps:

1. **Preprocessing**: Creation of equal-sized square patches from the whole image, which enables the extraction of local information.
2. **Feature Extraction**: Computation of Sign Binary Patterns (SBP) and Magnitude Binary Patterns (MBP) from each patches, using the CLBP algorithm, which defines a neighborhood based on a radius and number of neighbors.
3. **Feature Encoding**: Creation of a feature vector based on histograms of SBP and MBP for each patches, providing a simpler representation of the image.
4. **Classification**: Comparison of the feature vector to others, leading to the final classification of the image.

The results obtained from the PCLBP algorithm present an accuracy of 76% and a F-measure of 75%. It is a noteworthy contribution to the field, despite not surpassing the accuracy achieved by a previous study in 2008 [4]. However, it is important to highlight that the database used in this study contained a significantly larger number of images, 1043 in total, and has been utilized in other subsequent studies, as [6] and [7].

1.2.3 Convolutional Deep Belief Network (May 2018)

The study conducted in [6] was published in May 2018. The dataset used was the same than [5]. The method employed in the study was divided into four steps:

1. **Preprocessing**: The images were preprocessed using the Whitening method. This method was used to equalize the variance to the identity without having correlated intensity values.
2. **Feature Extraction**: In this step, a Convolutional Deep Belief Network (CDBN) was employed. A CDBN is a deep neural network that involves stacking multiple Convolutional Restricted Boltzmann Machines (CRBM) in a hierarchical manner [8]. Three CRBM layers were used in the study, with intercalated max-pooling to improve results.
3. **Feature Encoding**: A K-Means algorithm was applied to the features, and the centroids were used to form a codebook. An histogram of the formed code words was then used as the signature of the image.
4. **Classification**: A Support Vector Machine (SVM) was used for the final classification.

The study achieved an accuracy of 84% and an F-measure of 86%, which are better results compared to the previous study discussed [5]. These results indicate that the proposed method is effective in classifying histopathological images of neuroblastoma.

1.2.4 SIFT/SURF for Feature Encoding (November 2020)

The dataset used in [7] is the same as [5] and [6]. Authors note a challenge with this dataset. In fact, it is imbalanced, with one of the classes representing 50% of the data. The method is composed of 5 steps:

1. **Preprocessing**: The images in the dataset were cropped to 300×300 pixels.
2. **Feature Extraction**: Features were extracted from the images using Scale-Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) algorithms.
3. **Class Balancing**: Downsampling of the extracted features corresponding to the over-represented class.
4. **Feature Encoding**: The remaining features were encoded using the Bag-of-Visual-Words (BOVW) representation combined with the K-means clustering algorithm.
5. **Classification**: A SVM was used for the final classification.

The study found that using SIFT and SURF together did not result in a significant improvement in accuracy compared to using only one of the algorithms. However, using resampling techniques such as Synthetic Minority Over-sampling Technique (SMOTE) or NearMiss improved the accuracy of the models by making them more robust.

The best results obtained were 90% accuracy and 88% F-measure. These findings are comparable the outcomes observed in prior research.

1.2.5 Recent work related to Neuroblastoma

Classification with all the characteristics of the Shimada System (October 2022)

In [9] a pre-trained Hover-Net model on the PanNuke database is used to accurately segment cells in histological images. Subsequently, various morphological features were extracted for each nucleus, including area, perimeter, convex area and eccentricity. These data were combined with patient-related information such as sex, age, and the nuclear degradation rate in tumor cells (called Mitosis-Karyorrhexis Index (MKI) [10]). Subsequently, classification algorithms like Random Forest and K-NN were trained to classify patients into two groups: high-risk and low-risk. The best model classifies with an accuracy of 86.5%.

New characteristics for the Shimada System (January 2023)

In [11] three parameters were used: image entropy (S), fractal dimension (FD), and lacunarity (λ). Image entropy measures the complexity or randomness of the image. Fractal dimension characterizes the roughness or irregularity of the structures within the image. Lacunarity measures the gappiness or heterogeneity of the image texture. These parameters were combined into a machine learning algorithm and reached an accuracy of 82% to detect tumor malignancy.

The authors recommend to incorporate S, FD and λ features alongside the two primary ones, namely "Grade of Differentiation" and "MKI" (see Figure 1.1) for a more robust and refined classification.

1.3 Image Preprocessing for CNN Applications

1.3.1 Image normalization

Before delving into the details of image preprocessing methods for convolutional neural networks (CNNs), it's important to note that the calculations are performed on the entire image dataset, and by data point one refers to a single pixel from all the images in the dataset. The purpose of these preprocessing methods is to transform the image data in a way that allows the CNN to learn meaningful patterns and improve the accuracy of its predictions. Widely used methods include Mean normalization or Z-score normalization. By considering the entire dataset, rather than individual images, these methods aim to provide a comprehensive and representative view of the image data, leading to better results in the training and evaluation of CNN.

Mean normalization

Mean normalization is a preprocessing step that involves subtracting the average value of the data from each data point. This is done to center the data around the origin, reducing the impact of outliers and helping the CNN better learn the true patterns in the data. The mean is calculated over all the data points in the dataset.

Z-score normalization

Z-score normalization, also known as standardization, is a preprocessing step that involves transforming the data so that it has a mean of zero and a standard deviation of one. This is done by subtracting the mean and dividing by the standard deviation of the data. This method is useful for transforming data that has a skewed distribution, making the data more symmetrical.

1.3.2 Class imbalance

Class imbalance refers to a situation in which the number of samples in one class is vastly differ from those in another class. This leads to difficulties in training a model to accurately classify objects in the underrepresented class. To address this issue, common approaches are outlined in [12].

At the data level, two methods directly modify the dataset. One is *resampling*. It involves either oversampling the minority class or undersampling the majority class to balance the distribution of classes. Another solution is *synthetic data generation*, which generates additional synthetic samples for the minority class to increase its size.

At the algorithm level, two methods are commonly used during the learning process. The first is *cost-sensitive learning*, in which the algorithm is trained with the cost of misclassification for each class, reducing the impact of class imbalance. For example the cost function named Focal Loss adjusts the weighting of each sample based on the model's prediction confidence to put more focus on hard-to-classify samples. The second method is *ensemble learning*, which combines multiple classifiers to make a final prediction, utilizing a diverse set of classifiers to reduce the imbalance's impact.

Finally, hybrid methods combine multiple approaches to further mitigate the effects of class imbalance.

1.3.3 Image Augmentation

Image augmentation is an important technique in medical imaging for training CNN as it increases the size and diversity of the training dataset to prevent overfitting. In medical imaging, where data is limited and the subject matter is complex, image augmentation helps the model learn to identify relevant features and patterns, leading to more accurate and reliable predictions. The most widely used are geometrics and includes rotations, translations, flips, crops, skews, affine and elastic transformations. These methods are often used in combination to achieve varied and more complex image augmentations.

According to [13] these data augmentation methods belong to the model-free category, which do not require any learning models. There are two other categories, the first being model-based, which rely on neural network models, and the second being optimizing policy-based, which are algorithms based on reinforcement and adversarial learning. However, the latter two categories are much more complex to use, and require manual control afterwards in specific applications such as biomedical imaging.

1.4 CNN Architecture

1.4.1 Convolution types

Released on December 2022, the Zewen Li-s survey [14] provides the currently used types of convolutions in deep learning, each with its own unique properties and applications.

- **Classical Convolution** involves adding zero padding to the input data to prevent information loss at the edges, and using strides to control the density of the output feature map. The dilation parameter is used to expand the local region studied by the convolution operation.
- **Deformable Convolution** is designed to increase the robustness of the network to geometric transformations in the input data.
- **Group Convolution** allows for wider networks with fewer parameters per layer or fewer layers with the same computational cost.
- **Steerable Convolution** is designed to adapt to specific lines or shapes in the input image.
- **Graph Convolutional Network** performs convolutions on graph-structured data, making them well suited for processing complex relationships between nodes in graph-based data.

1.4.2 CNN layers

The architecture of a CNN is composed of several layers, each of which performs a specific task in the process of image classification. Four different layers are found in a typical CNN:

- **Convolutional Layer**: performs the convolution operation, which involves element-wise multiplication of the inputs and a set of learned filters, followed by a summation operation. The filters act as feature detectors, capturing important features from the input image. The output is called a feature map.
- **Pooling Layer**: down-samples the feature map, by performing operations such as max-pooling, in which the maximum value in each region of the feature map is taken as the output. This operation improves computation efficiency and reduces the risk of overfitting.
- **Activation Layer**: introduces non-linearity into the CNN. It allows the network to capture complex relationships.
- **Fully Connected Layer**: standard feed-forward neural network layer, which performs the final classification.

The position of Pooling and Activation Layers is not universal and depends on the design of the network. In some cases, it may be preferable to omit these layers to achieve better results. Another useful technique for improving the performance is *dropout*, which involves randomly disabling some of the neurons or channels during training to prevent overfitting [15].

1.4.3 Activation functions

Activation functions are used to introduce non-linearity into the network. Without its, the network would only be capable of linear transformations, making it difficult to model complex data distributions. Activation functions have different properties and can be suited to specific tasks and architectures. For example, the sigmoid activation function is well-suited for binary classification problems, while the softmax activation function is commonly used for multi-class classification problems. The table 1.1 summarizes the classic and recent activation functions used in current CNNs.

1.4.4 Backbone

Backbones are architectures that form the main part of networks for extracting features from raw images, which are then used for classifying. Backbones have evolved to become increasingly deep and complex, with architectures such as VGG [16], ResNet [17], and Inception [18] being widely adopted for computer vision tasks. Pre-trained backbones are often used to perform transfer learning, allowing computer vision tasks to be achieved with a small database.

The backbone version often refers to the number of layers. The deeper versions tend to get better results, but also require more computational resources. Three common backbone families are detailed in the following points.

Function	Strengths	Weaknesses
Linear	Simplicity, easy to compute	Can not solve complex problems
Sigmoid	Suitable for binary classification problems	Slow convergence, poor performance
Softmax	Outputs probability for multiple classes	Sensitive to extreme value inputs
Tanh	Zero-centered output, widely use	Prone to oscillation, vanishing gradient
ReLU	Good stability, fast, avoid vanishing gradient	Can result in network learning nothing (Dying ReLu problem)
Leaky ReLu	Avoid Dying ReLu Problem	Hyper-parameter (0.02 to start)
PReLU	Learn the Leaky ReLu hyper-parameter	Prone to oscillation
ELU	Good accuracy, avoid Dying ReLu Problem	Computationally expensive
Swish	Good accuracy, avoid Dying ReLu Problem	New: not as well understood as others
Mish	Outperforms Swish and ReLU across common standard architectures	New: not as well understood as others

Table 1.1 – Activation functions for CNN layers

Visual Geometry Group

Visual Geometry Group (VGG) repeats small (3x3) convolution blocks several times and stack them on top of each other. The dimension reduction is performed by max pooling layers: it keeps a single maximum value for an entire window. An illustration of VGG-16 architecture is provided in appendix A.1.

The last layer of the VGG architecture is a fully connected with 4096 inputs and 1000 outputs for the 1000 ImageNet classes. The classification part is independent of the versions and consists of 3 fully connected layers. Only the number of convolutional layers becomes more important as the versions are higher. VGG is available in four versions: 11, 13, 16, and 19. The relationship between the version v and the number of convolutional layers c can be expressed as $v = c + 3$.

Residual Network

Residual Network (ResNet) uses basic modules that contain two parallel branches. The main one performs a linear transformation and the residual one adds the original input to the output of the main branch. This avoids the problem of the vanishing gradient that occurs when the depth of the network is increased.

The last layer of the Residual Network architecture is a fully connected layer that varies in the number of inputs. For versions 18 and 34, it has 512 inputs, while for versions 50, 101, and 152, it has 2048 inputs. This layer also has 1000 outputs and a bias term. The 1000 outputs correspond to the probability of each of the 1000 classes in the ImageNet dataset.

Inception

Inception combines several different types of convolution blocks into a single module. These blocks include convolutions of different filter sizes (1x1, 3x3, 5x5) and pooling operations. The objective is to allow the network to capture patterns at different spatial scales. There are several versions of Inception, ranging from Inception v.1 to Inception v.4. An illustration of the architecture of Inception v.3 is given in appendix A.2. The final layer is a fully connected with 2048 inputs and 1000 outputs.

Performance on ImageNet

Table 1.2 presents the performance of the backbones discussed on ImageNet. The weights producing these results are available online with PyTorch [19]. Acc @5 denotes the accuracy of accurately identifying the class within the first five predicted categories.

Backbone	Acc @1	Acc @5	#Params	FLOPs
ResNet-18	69.76%	89.08%	11.7M	2B
ResNet-34	73.30%	91.42%	22M	4B
ResNet-50	76.15%	92.87%	26M	4B
ResNet-101	77.37%	93.56%	45M	8B
ResNet-152	78.31%	94.06%	60M	12B
VGG-11	69.02%	88.63%	133M	8B
VGG-13	69.93%	89.25%	133M	11B
VGG-16	71.59%	90.38%	138M	15B
VGG-19	72.38%	90.88%	144M	20B
Inception v.3	77.29%	93.45%	27.2M	5.71B

Table 1.2 – Backbone performances on ImageNet

1.5 Optimization

1.5.1 Architecture Scaling

When scaling a CNN, the terms depth, width and resolution are used to describe different aspects of the network architecture. Here is an explanation of each term:

— **Depth**: is the number of layers in the network. Increasing the depth improves the performance by using more hierarchical features.

— **Width**: is the number of neurons or filters in a given layer of the network. Increasing the width improves the performance with more accurate features.

— **Resolution**: is the size of the input images. Increasing the resolution provides more details and allows more complex representations.

In general, when scaling a CNN, a trade-off must be made between the representational capacity of the network and its computational cost and the risk of overfitting.

Architecture scaling is a method for dealing with this compromise and effectively increasing the dimensions of a functional CNN. This approach is particularly useful in scenarios where more data becomes available or computational capabilities increase, making it feasible to consider more complex networks without starting again from scratch. Alternatively, intentionally designing a smaller model that can be fine-tuned quickly and then using the formulas proposed in [20] to increase its dimensions is also a viable strategy.

1.5.2 Loss functions

Loss functions, also known as cost functions, measure the difference between the network's predicted output and the true label, providing a signal to the network to adjust its weights and biases. The goal of training is to minimize the loss function, effectively reducing the discrepancy between the predicted and true labels. The choice of loss function will depend on the specific task and architecture of the network. In [21], authors divide loss functions for classification problem in two: margin-based and probabilistic.

Margin-based losses separate the true label from the incorrect labels by a certain threshold called margin. There are essentially designed for binary classification purpose. In the other hand, probabilistic losses are based on the concept of maximum likelihood estimation, which involves finding the values of the network parameters that maximize the probability of the true label given the inputs.

The table 1.3 provides the mainly used loss functions for classification problems based on [14] and [21]. The first group corresponds to margin-based, and the second one to probabilistic functions.

1.5.3 Optimizers

The table 1.4 is extracted from [14] and provides the currently used optimizer for CNN.

Early stopping is the commonly used method to know when the optimization is complete. The main idea is to monitor the model's performance on a separate validation set during training, and stop it when the performance begins to deteriorate.

Function	Strengths	Weaknesses
Cosine Similarity Loss	Simple	Not designed for image application
Hinge Loss	Robust to outliers	More used with SVM than CNN
Large-margin Softmax Loss	Design for inter-class separability	Sensitive to feature scales
Cross Entropy	Commonly used for multi-class	Sensitive to class imbalance
Focal Loss	Focus on hard examples	Increase complexity
Constrative Loss	Efficient with dimensional reduction	Face recognition applications
Center Loss	Model within-class compactness	Hyperparameters

Table 1.3 – Loss functions for classifications

Optimizer	Strengths	Weaknesses
SGD [22]	Excellent generalization	1. Prone to saddle point or local optima 2. Slow convergence rate 3. Sensitive to HP initialization and learning rate selection
SGD Momentum [23]	1. Accelerate in directions of steady descent 2. Not prone to failing into local optima	Sensitive to HP initialization and the values of learning rate and momentum
Adagrad [24]	1. Adjust learning rate adaptively 2. Convergence rate is faster on data with sparse features	1. Worse generalization and likely to converge to sharp minimum 2. Gradient vanishing
RMSprop [25]	1. Adjust learning rate adaptively but less aggressive than Adagrad 2. Convergence rate is faster on data with sparse features 3. Built-in Momentum	Worse generalization and likely to converge to sharp minimum
Adam [26]	1. Automatically decay the learning rate 2. Convergence rate is faster on data with sparse features 3. Combine the advantages of Momentum, Adagrad and RMSprop	Worse generalization and likely to converge to sharp minimum

Table 1.4 – Optimizers for CNN architectures

In detail, the model is optimised on a training set, and at the end of each epoch, the model's performance is evaluated on a separate validation set that has not been used during optimisation. If the performance on the validation set stops improving for a certain number of epochs, then the training is stopped, and the model is returned to the last epoch where the performance on the validation set was the best.

1.6 Learning

Over the years, CNN has become increasingly effective on classic evaluation/competition databases such as ImageNet or COCO. This progress can be attributed to the implementation of more complex mechanisms, and these advancements also require an increasing number of parameters to be trained. This increase in parameters leads to a growing need for labeled data. However, this presents a challenge in the context of biomedical applications where data is scarce and publicly available in limited amounts.

1.6.1 Transfer Learning

Transfer Learning involves taking a pre-trained neural network, typically trained on a large dataset such as ImageNet, and fine-tuning it on a new, and smaller dataset. The idea behind this approach is to leverage the knowledge learned by the pre-trained network and apply it to the new task. Thus reducing the amount of data and computational resources required to train a network from the beginning.

One example of transfer learning is the use of a CNN architecture. In computer vision, CNN has proven to be highly effective in tasks such as image classification and object detection. A common approach to transfer learning is to use a pre-trained CNN as a backbone. It means a fixed feature extractor, where the weights are not updated during the fine-tuning process. For example in [27], several popular CNN architectures, including DenseNet201, VGG16, Xception, and Inception v.3, were evaluated as backbones for a multi-classification task on histopathological images of breast cancer.

Another approach of transfer learning with CNN is fine-tuning the weights of the pre-trained network. In this case, a small learning rate is used to update the weights of the pre-trained network, taking into account the new task. The goal is to make small adjustments to the weights of the pre-trained network that are suitable for the desired classification.

1.6.2 Active Learning

Active learning is a machine learning technique that focuses on the process of training a model by actively selecting data samples from a large pool of available samples. The goal of active learning is to minimize the amount of labeled data needed to train a model to a desired level of accuracy, while maximizing the accuracy of the model.

In traditional machine learning, a large dataset is randomly split into a training set and a test set, with the training set being used to train the model and the test set being used to evaluate the performance of the model. However, in many cases, the amount of labeled data is limited, which can lead to poor model performance. Active learning addresses this issue by allowing the model to actively select the most informative samples to be labeled and added to the training set. This approach was successfully used in [28] to develop a classifier of Osteosarcoma histopathological images (related to a bone cancer).

1.7 Conclusion

In summary, Neuroblastoma Classification encompasses two distinct categories: favorable histology (FH) and unfavorable histology (UH). A key factor in this classification is the Grade of Differentiation exhibited by the tumor cells. Three stages can be identified: Differentiated, Poorly Differentiated, and Undifferentiated, which are closely associated with the severity of the disease. This classification primarily relies on the morphology and distribution of the tumor cells in the histological Whole Slide Images (WSI). Four algorithms have been reviewed, and their respective performance outcomes are summarized in Table 1.5.

Paper (reduced title)	**Date**	**Framework**	**Accuracy**
Computer-aided Evaluation on WSIs [4]	10/08	Cytoplasm and neuropil segmentation, feature extraction, voting procedure	88%
Patched Completed Local Binary Pattern is an Effective Method [5]	04/18	Patch, binary pattern, histogram for encoding images	76%
Convolutional Deep Belief Network (CDBN) with Feature Encoding [6]	05/18	Convolutional Restricted Boltzmann Machines	84%
Classification of Histopathological Images Using Machine Learning [7]	11/20	SIFT/SURF features, SVM	90%

Table 1.5 – Algorithms classifying the Grade of Differentiation in histological images

Preprocessing techniques were employed to prepare the dataset, including cropping, normalization, and selecting a color space. Feature extraction was primarily carried out using well-defined methods, although the CBDN study used a neural network to extract more complex features without the need for handwritten functions. Finally, classification was mostly performed using clustering methods such as K-means and Support Vector Machines.

We also reviewed Convolutional Neural Network (CNN) methods used in preprocessing, activation functions, and optimization functions, along with their strengths and weaknesses. The advantages of using a CNN consist in both feature extraction and classification can be carried out in a single block of automatic learning, allowing for the utilization of a large amount of data, resulting in improved generalization.

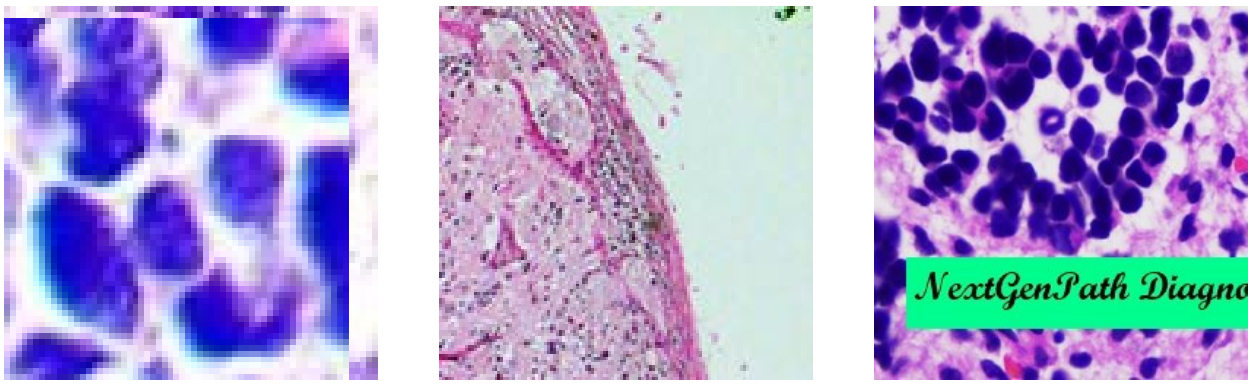
Chapter 2: Database

The database was used for the first time during this project. It therefore has a number of unkown imperfections or non-standard properties, and some preprocessing is required. The objective of this chapter is to form three independent datasets where all classes are represented. These datasets are called «training», «validation» and «test» and should partition the set of images with the following ideal proportions: 70%, 15% and 15%. The training dataset is used to learn the weights, while the validation dataset to supervise this training. The test dataset is only involved when the model is trained and allows its performance to be evaluated independently of the data used during training. Note that the terminology «validation» and «test» are often reversed from one scientific paper to another.

2.1 Database Specifications

2.1.1 Image regularity

Some images have watermarks, borders or other distinctive features not related to the nature of the disease being studied. Figure 2.1 shows some of these irregularities that need to be removed.



(a) Vignetting (b) Non biological tissue area (c) Watermarks

Figure 2.1 – Examples of irregularities encountered

The images made available have fluctuating dimensions: ranging from 321×241 pixels to 4000×3000 pixels, which is not standard for a convolutional network training.

2.1.2 Number of images

In our application framework, few labelled images are available. This is due to the need for an expert in the field to study the images, which are often more complex and therefore more time-consuming than differentiating a dog from a cat. In addition, special permissions may be required to publish such images online, especially when they are related to the human body or children.

The database is also subject to class imbalance, with a factor of 2 between the most represented class and the least represented one. This can lead to poor learning and more difficult to read performance. For example, consider an exaggerated case: two classes, a 95% healthy patient class and a 5% sick patient class. A model that systematically predicts the majority class has a performance of 95%, whereas it is not a «good» model in the sense that it never detects the disease.

2.2 Image Size

2.2.1 Image size for CNN applications

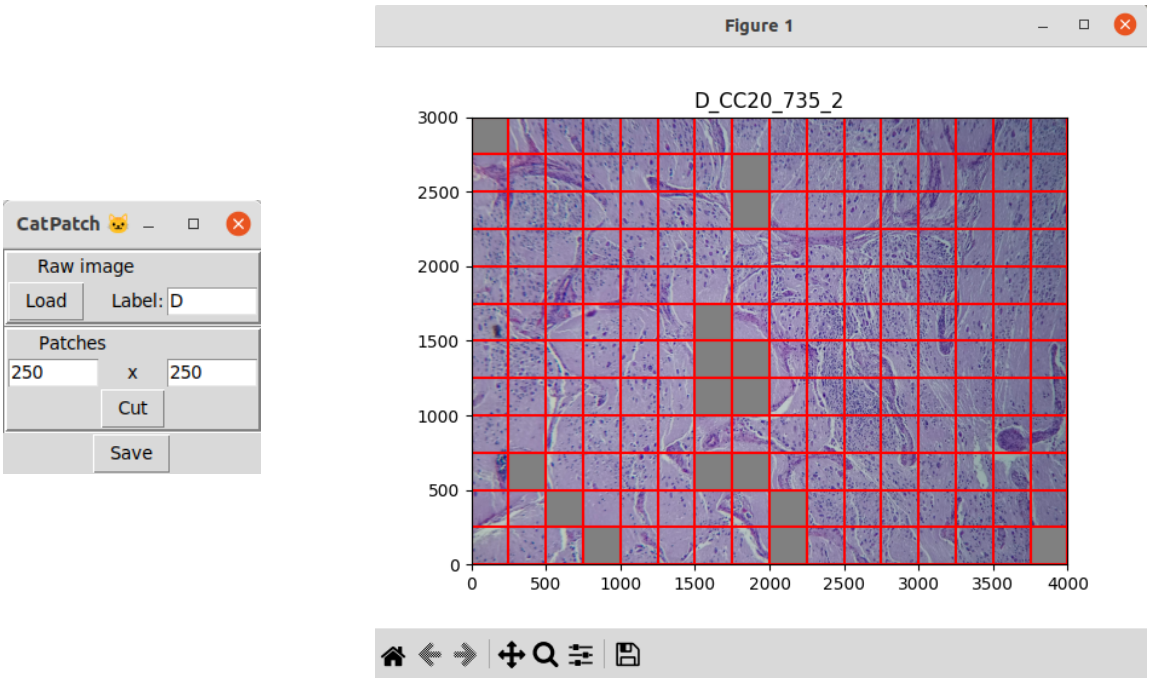
In the context of a Convolutional Neural Network (CNN) application, it is essential for the input images to have uniform dimensions. Typically, popular networks like VGG, ResNet, and DenseNet utilize an input size of 224×224 pixels. This particular size was chosen due to its divisibility into 32×32 pixel squares, which facilitates efficient parallel processing on GPU architectures. Moreover, this size is generally sufficient for capturing the requisite patterns required for recognition. However, it should be noted that this is not an absolute rule, as the Inception architecture, for instance, employs a format of 299×299 pixels.

In Chapter 4, the VGG, ResNet, and Inception architectures are examined. To ensure consistency in the datasets and enable comparison of training sessions across different architectures, the images are cropped to dimensions of 250×250 pixels. For models using the 224×224 pixel format, a random crop is performed at each iteration to form an image batch for the given optimization process. For the 299×299 pixel format, resizing is achieved through interpolation. The choice of the 250×250 pixel format minimizes pixel loss since the patches do not overlap, and the dimensions of Whole Slide Images (WSIs) in our raw database are often multiples of 250.

A similar methodology was employed in a previous study [5], where WSIs were divided into patches. The use of cropping is preferred over large resizing as it allows for a significant increase in the number of patches without distorting the morphology or existence of the observed cells. Additionally, a pre-crop step is performed to eliminate artifacts and watermarks, as depicted in Figure 2.1.

2.2.2 Graphical Interface for an efficient cutting

The patches do not necessarily inherit the label of the original image. Indeed, if no cells are visible then it is preferable to do not keep the slicing as it decreases the data quality of the class involved. In order to increase efficiency, a graphical interface has been developed (figure 2.2). It allows the original image to be cut using a grid according to the patch size entered. The user can then click on certain boxes which will be greyed out (figure 2.2b). The greyed out areas correspond to the cuts that cannot inherit the label from the original image. Once this review is complete, the patches are created, named and placed in the correct labelled folder.



(a) CatPatch menu (b) Image greying out

Figure 2.2 – Graphical interface improving efficiency for patch labelling

The resulting cut-out helps to solve the problem of a small number of images while standardising the dimensions. However, this approach does have certain drawbacks. Specifically, not all pixels are preserved

since some patches do not inherit the labels and, as a result, are discarded. Additionally, the dimensions of the original images may not align perfectly with the patch dimensions.

As an example, in our dataset containing 61 original images of varying size totalling 116,077,948 pixels were available. After cutting out the viable areas that could retain the labels, 1,570 patches of size 250x250 were obtained. This represents a loss of 15.47% in terms of pixels. This loss is explained by the non-propagation of labels for 12.94% and by the non-usable borders for 2.53%.

2.3 Dataset Allocations

Once the sizing problem has been solved, a new one arises: the allocation of cuts to the datasets. As a reminder, the *training*, *validation*, and *test* datasets must be independent. It is therefore impossible to allocate two patches from the same medical case to two different datasets. A second constraint is the respect of the proportions of the different datasets (approximately a 70%, 15%, 15% partitioning).

2.3.1 Problem Statements

The objective is to partition a set of patches into 3 groups P_1, P_2 and P_3. This means that the proportions p_1, p_2 and p_3 associated with the partitions P_1, P_2 and P_3 must sum to 1. One sets $p_1 > p_2 \geq p_3$. A key assumption of the problem is that $p_1 > 0.5$. Indeed, a partition must contain at least half of the available pixels, and will be associated with the training dataset.

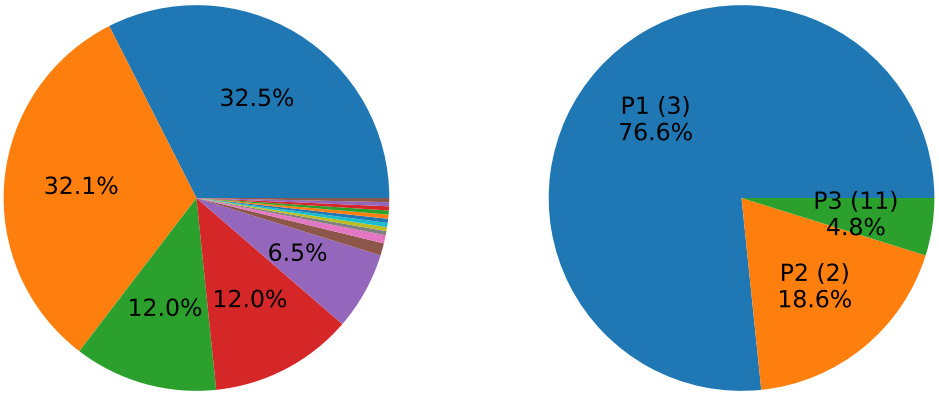
A family of patches is associated to each image. A family has three properties:

1. It groups together only the viable patches with sufficient information to inherit the original label.
2. It is measured by the total number of pixels that its elements contain.
3. It is non-separable: all its elements must belong to the same partition.

2.3.2 Proportional and Sequential Partitioning

A simple method is then to sort the families of the different classes according to their weight. Then allocate the most important families of the same class to P_1 until they exceed p_1, and repeat this process for all classes. Once P_1 is filled, the families of P_2 are allocated by the same process with the remaining important families until exceeding p_2. And P_3 groups the last families, those with the least pixels.

In practice, this distribution has two major defects. The P_1 and P_2 partitions present a lack of diversity and P_3 do not collect enough images, because P_1 and P_2 concentrate all the important families. This phenomenon is particularly accentuated in the case where few images are available and some of them are particularly large. This is illustrated in figure 2.3. For a given class, figure 2.3a gives the proportion of images in terms of useful pixels (from the label inheritance cut), and figure 2.3b the resulting distribution with the number of images in parenthesis. In pairs, P_1 and P_2 combine 5 families while P_3 contains 11.



(a) 16 families ordered by pixel proportions

(b) Proposed partitions (for a 70-15-15 target partitioning)

Figure 2.3 – Proportional and Sequential Partitioning

2.3.3 Additional Statement: Diversity

A new constraint emerges: that of respecting an equidistribution in the number of families constituting the partitions. More generally, the partitioning must be evaluated on two metrics:

1. The respect of the target proportions p_1, p_2, p_3 (relative to the number of pixels)
2. The respect of a target diversity d_1, d_2, d_3 (relative to the number of families)

Target diversities d_1, d_2 and d_3 are assigned to P_1, P_2, P_3 and give the share of the number of families assigned. So, $d_1 + d_2 + d_3 = 1$, and in our application framework $d_1 = d_2 = d_3 = 1/3$ because diversity is important at all levels: during training to generalise well and during evaluation to reduce uncertainties on performance.

2.3.4 Iterative Filling Partitioning (1D)

One solution for the diversity problem is to avoid filling P_1 completely from the start, in favour of a more iterative approach named Iterative Filling Partitioning (IFP).

Initialisation

First, families that can not be assigned to P_2 and P_3 because they are too large are assigned to P_1. If N_1 families are assigned to P_1 then P_2 and P_3 are deficient in $N_2 = \frac{d_2}{d_1} \times N_1$ and $N_3 = \frac{d_3}{d_1} \times N_1$ families respectively. N_2 and N_3 are rounded to the nearest whole number, or equal to N_1 in our application framework. Then, in turn, the remaining families with the highest weight are assigned, if possible, to P_2 and P_3. If a family cannot be assigned, it is ignored and will be assigned later to P_1.

Iterations

The partitions are then alternately assigned a family. The largest remaining family is always select first. Partitions P_2 and P_3 are considered filled when no remaining family can prevent them from exceeding the proportions p_2 and p_3, or when all families have been assigned. In the end, any remaining families are assigned to P_1.

Initialisation parameter

The IFP method is sensitive to the initialization where P_1 is filled with N_1 images. This is why a parameter μ is introduced such that N_1 is the number of families gathering a proportion of total pixels in the class greater than or equal to $\mu \times min(p_2, p_3) = \mu \times p_3$. This parameter μ belongs to $]0,\ 1]$ and allows P_1 to absorb the families saturating too quickly the partitions P_2 and P_3, preventing them from reaching their target diversity.

Loss function

An exhaustive search for the optimal μ^* is performed. Let the proportions p_i and the diversities d_i effectively obtained be noted with an eff power.

$$D(\mu) = \frac{1}{6}\left[\sum_{i=1}^{3} |p_i - p_i^{eff}(\mu)| + \sum_{i=1}^{3} |d_i - d_i^{eff}(\mu)|\right] \tag{2.1}$$

$$\mu^* = \underset{\mu \in [0,1]}{\operatorname{argmin}}\ D(\mu) \tag{2.2}$$

This metric is equivalent to averaging the differences reached on both proportions and diversities.

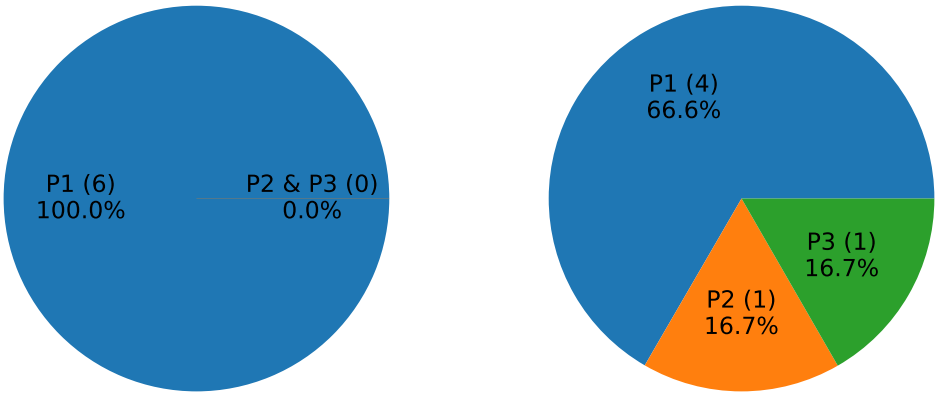
The obtained partitions on the example given in Figure 2.3 are $(72.2\%, 14.1\%, 13.7\%)$ with a diversity of $(0.38, 0.31, 0.31)$. The target parameters were $(70\%, 15\%, 15\%)$ and $(1/3, 1/3, 1/3)$. As there is 16 families, which is not divisible by 3, the reached diversity is optimal.

The IFP solution is called 1D because only one parameter is used, and results in the minimisation of a curve. The modification proposed in the next section introduces a second one. It is a 2D solution.

2.3.5 Iterative Filling Partitioning with Overfill (2D)

Highlighting the IFP Failure in a Simple Case

The 1D solution does not allow the P_2 and P_3 partitions to exceed their proportions. Consider the following case: 6 families of equal weight with the same target parameters: $(p_1, p_2, p_3) = (0.7, 0.15, 0.15)$ and $(d_1, d_2, d_3) = (1/3, 1/3, 1/3)$. The application of the 1D solution gives the partition shown in Figure 2.4a. A more intuitive and better minimising partition for equation 2.1 is given by figure 2.4b but implies that P_2 and P_3 must exceed their respective capacities.



(a) IFP partition (b) Best partition

Figure 2.4 – Simple case highlighting the need for P2 and P3 to exceed their capacities

Proposed Solution

By allowing P_2 and P_3 to exceed their capacity of $\varepsilon = 0.02$ once the initialization with the assignment of N_1, N_2 and N_3 families is completed, the problem is solved (Figure 2.4b). The choice of ε depends on the distributions, therefore it is a new and independent parameter to search for:

$$D(\varepsilon, \mu) = \frac{1}{6}\left[\sum_{i=1}^{3} |p_i - p_i^{eff}(\varepsilon, \mu)| + \sum_{i=1}^{3} |d_i - d_i^{eff}(\varepsilon, \mu)|\right] \tag{2.3}$$

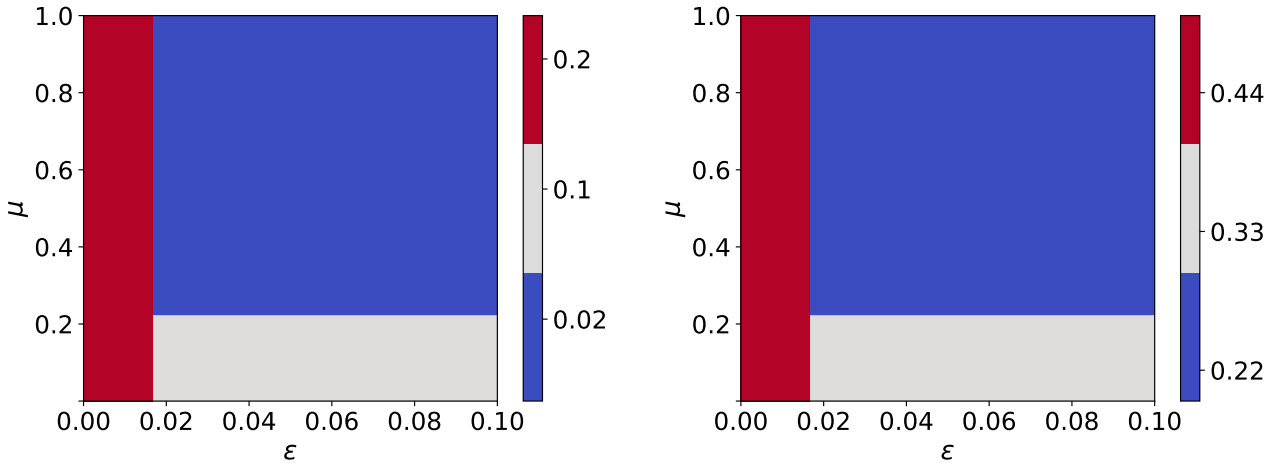
$$(\varepsilon^*, \mu^*) = \underset{\substack{\varepsilon \in [0,1] \\ \mu \in [0,1]}}{\operatorname{argmin}} D(\varepsilon, \mu) \tag{2.4}$$

Heatmaps

The search for the pair (ε^*, μ^*) results in minimising a heatmap. A heatmap is an image where the pixel color represents the loss for a given coordinates (ε, μ). The colder the temperature, the more blue the pixel colour tends to be: this is where minima are found. Here, two heatmaps are used. The first one is related to the respect of capacities, given in figure 2.5a and the second is related to the respect of diversity, given in figure 2.5b.

With the simple illustrative example of the Figure 2.4, the regions of the maps are identical. The minimization process is conducted on the cumulative values of both maps. Implying that a 1% disparity in capacity holds equivalent significance to a 1% divergence in diversity. This equivalence may be unreliable, and can be easily balanced by assigning weights to the maps if needed.

It is important to note that IFPO is necessarily better or equivalent to IFP because IFP is a special case of IFPO where $\varepsilon = 0$. On figures 2.5, $\varepsilon = 0$ corresponds to the red area. The 2D search allows more freedom, and two better areas in white and blue appears with a key point (0.017, 0.22). The abscissa $\varepsilon = 0.017$ corresponds to an additional 1.7% for the capacity of P_2 and P_3, leading to $15\% + 1.7\% = 16.7\% = 1/6$: the minimal and required capacity for a partition to contain a family.



(a) Loss Capacity

(b) Loss Diversity

Figure 2.5 – Heatmaps of the IFPO method (from the case presented in Figure 2.4)

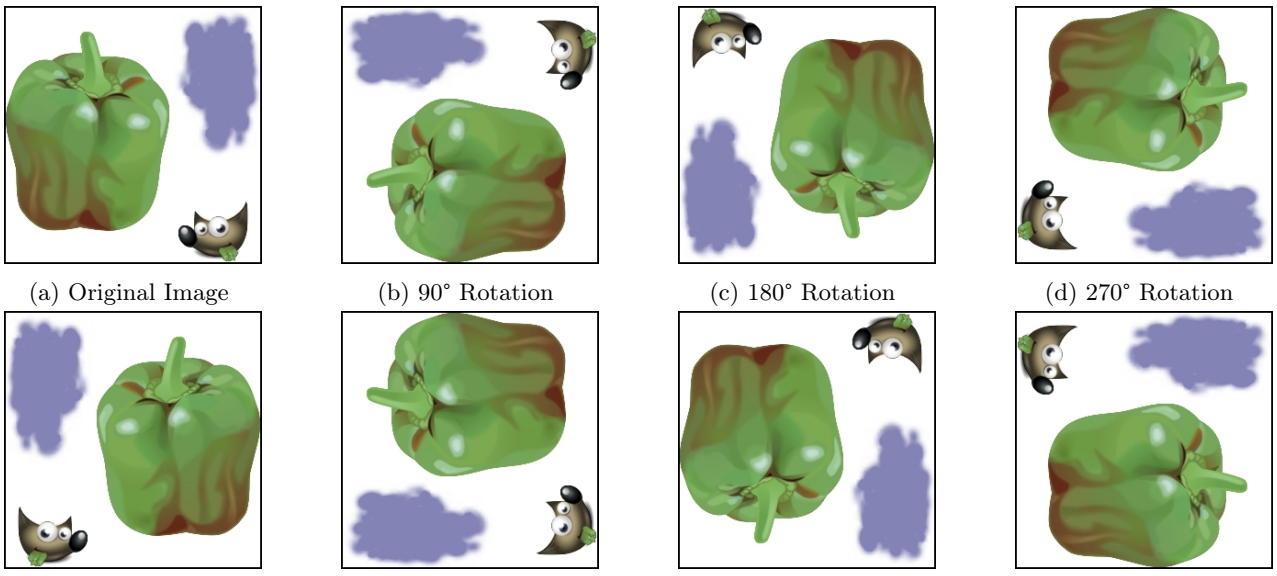
2.4 Class Imbalance

A class imbalance occurs when a dataset has one of its classes under-represented in terms of number of images. This leads to generalisation problems during training and makes it difficult to interpret certain metrics during the evaluation phase. Two strategies exist: subsampling the over-represented classes or artificially creating new images for the minority classes.

2.4.1 Global Geometric Transformations

In our application framework, few images are available, so augmentation methods are preferred. Only global geometric transformations (rotation, flip) are considered, and not transformations involving local geometric deformations in order to preserve the characteristics of the cells.

For square images, there are 7 transformations that do not alter the internal geometries. They are illustrated in Figure 1 and are made up of two elementary transformations: a 90° rotation and a flip.



(a) Original Image

(b) 90° Rotation

(c) 180° Rotation

(d) 270° Rotation

(e) Original Flipped

(f) Flipped - 90° Rotation

(g) Flipped - 180° Rotation

(h) Flipped - 270° Rotation

Figure 2.6 – 7 Global Geometric Transformations

2.4.2 Strategy for Assigning Transformations

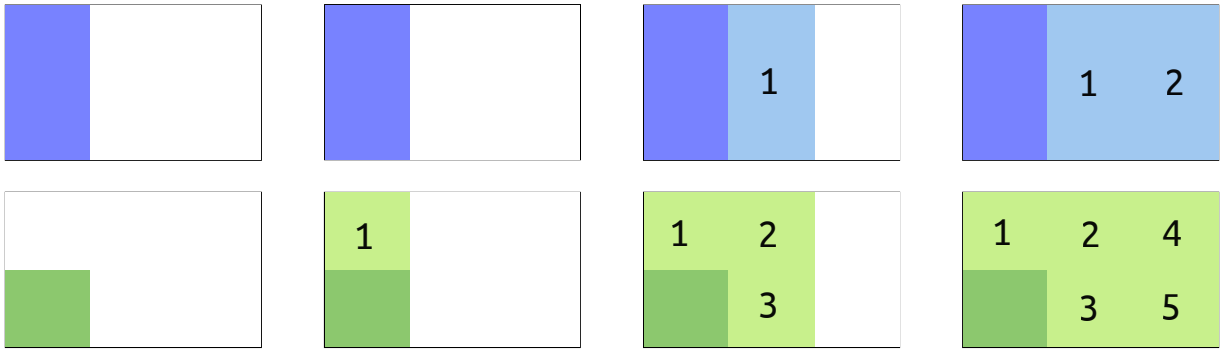
The challenge is twofold: solving the imbalance class problem with a number of transformations, and if possible increasing all classes with the remaining transformations. To achieve this, the two extreme classes must first be considered: the one with the fewest images and the one with the most images. The ratio of the number of images rounded to the nearest integer is noted as $r \geq 1$.

Special case $r = 1$

If $r = 1$, there is no real class imbalance. The rounding in the definition of r may hide that there are not exactly the same number of patches, but this does not justify the privatization of a transformation for a given class. It is better to collectively increase the amount of samples in this case.

Case $r = 2$

If in the end n remaining transformations are applied to the majority class, then 2n+1 transformations must have been provided for the minority class which is down by 1 to 2 patches. This relationship is illustrated in Figure 2.7. Two classes are shown, a minority class in dark green, and a majority class in dark blue. The lighter areas represent the proportion of generated samples from some numerated transformations of the darker area.



(a) Starting Situation (b) Balancing Point (c) 1st Augmentation (d) 2nd Augmentation

Figure 2.7 – Balancing and Augmentations

To balance the situation, only one transformation is necessary. This corresponds to the +1 factor. Then, to collectively increase the data, the factor 2 must be respected. This leads to $2n + 1$ transformations for the minority class.

Generalisation

Let r, the ratio between the two extreme classes. Then balancing requires $r - 1$ transformations. The collective increase takes place at a rate of 1 for r. Therefore, by noting n the number of transformations applied to the majority class, it is necessary to provide $r \times n + (r - 1)$ for the minority class.

For the intermediate classes, the same formula applies, only the coefficient r differs, which should necessarily be less than or equal to the one of the minority class.

2.4.3 Application

Let t the number of transformations. The relation 2.5 optimises the use of these transformations, with r the ratio of the minority class previously defined.

$$r \times n + (r - 1) \leq t \tag{2.5}$$

By isolating n, it can be deduced that all classes are minimally augmented by $\left\lfloor \frac{t+1}{r} \right\rfloor - 1$ transformations, with $\lfloor . \rfloor$ as the floor operator.

However, there is no available transformation for collectively increases all classes when $(t + 1)/r < 1 \Leftrightarrow t < r - 1$. Therefore the Number of Global Class Augmentation is given by the following expression:

$$NGCA = \begin{cases} \left\lfloor \frac{t+1}{r} \right\rfloor - 1 & \text{if } t \geq r-1 \\ 0 & \text{else} \end{cases} \tag{2.6}$$

2.4.4 Example

In the database provided, two classes are available: D and PD. The D class was in the minority with a ratio $r = 2$ on the 3 datasets. Balancing and augmentation was achieved with all $t = 7$ transformations for D and 3 transformations for the majority class PD. Table 2.1 gives the raw numbers of images before and after balancing and augmentation. Table 2.2 provides the ratio r before rounding to the nearest integer for the 3 datasets.

It is interesting to note that the majority and minority classes are reversed in the process. Furthermore, the unrounded r-ratio is exactly the same after balancing and augmentation than only with balancing. This is due to the fact that the NGCA definition (2.6) is optimal as the floor operation is redundant ($r = 2$ divides the integer $t + 1 = 8$).

Class	Starting Situation	Balancing Point	Final Augmentation
PD (train)	734	734	2,936
D (train)	420	840	3,360
PD (valid)	127	127	508
D (valid)	82	164	656
PD (test)	127	127	508
D (test)	80	160	640

Table 2.1 – Number of samples through transformations

Dataset	Starting Situation	Balancing Point	Final Augmentation
Train	1.75	1.14	1.14
Valid	1.55	1.29	1.29
Test	1.59	1.25	1.25

Table 2.2 – Unrounded r-ratio through transformations

2.5 Standardisation

The order in which class imbalance addressing and standardization should be performed lacks a universal rule. Nevertheless, it is typically advised to balance datasets before scaling them. This recommendation stems from the fact that achieving precise classification outcomes necessitates consistent application of scaling to all classes. This practice guarantees that features are scaled uniformly, which can prove crucial for various machine learning algorithms.

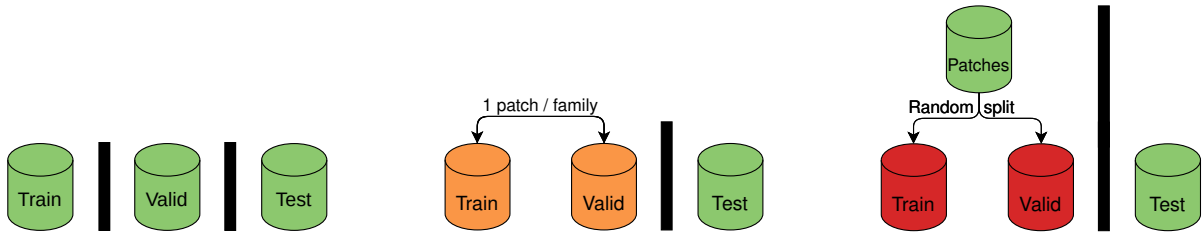
To standardize a given dataset of RGB images, the mean and variance are computed across the three color channels of each image. Each pixel value is then transformed by subtracting the mean and dividing by the standard deviation, ensuring that the dataset is centered around zero and has a standard deviation of one.

This operation is performed on training, validation and test datasets independently to ensure that each one is transformed to have a similar distribution. It also prevent information leakage between datasets, as standardization is not performed on the entire database as a whole.

2.6 Experiments on Dataset

In order to test the approach developed in this chapter, several experiments are performed. The objective is to study the behaviour of the training curves when the training and validation datasets are correlated. This correlation means that patches belonging to the same family are found in these two datasets. The test dataset always remains independent of the two others to allow to expose, when necessary, the training biases. Figure 2.8

illustrates the different shuffles that are performed in these experiments. It is important to note that the same architecture, the same optimisation algorithms and the same training parameters (learning rate, momentum, batch size, epochs) are used. Only the database partition is modified.



(a) Experiment 0 (b) Experiment 1 (c) Experiment 2

Figure 2.8 – Set of experiments conducted

In experiment 0, all datasets are independent. The goal is to show what happens with the training for the chosen model. This one is designed to not generalise on the data and to be able to memorise the samples of the training dataset.

In experiments 1, the datasets are more or less correlated, but remain relatively independent because the exchange involves a single patch per family and some families contain more than 150. The number of patches in each dataset remains unchanged after this exchange.

In experiment 2, the samples for the training and the validation datasets are randomly assigning. The correlation is significant and stronger than in Experiments 1.

Results

Each experiment was conducted ten times. Figure 2.9 depicts the mean loss across epochs for both training (dotted curves) and validation (plain curves). The magnitude of loss increases with greater independence between the datasets. Experiment 1 exhibited a similar pattern to Experiment 0 with a minor magnitude, indicating that some correlation is present.

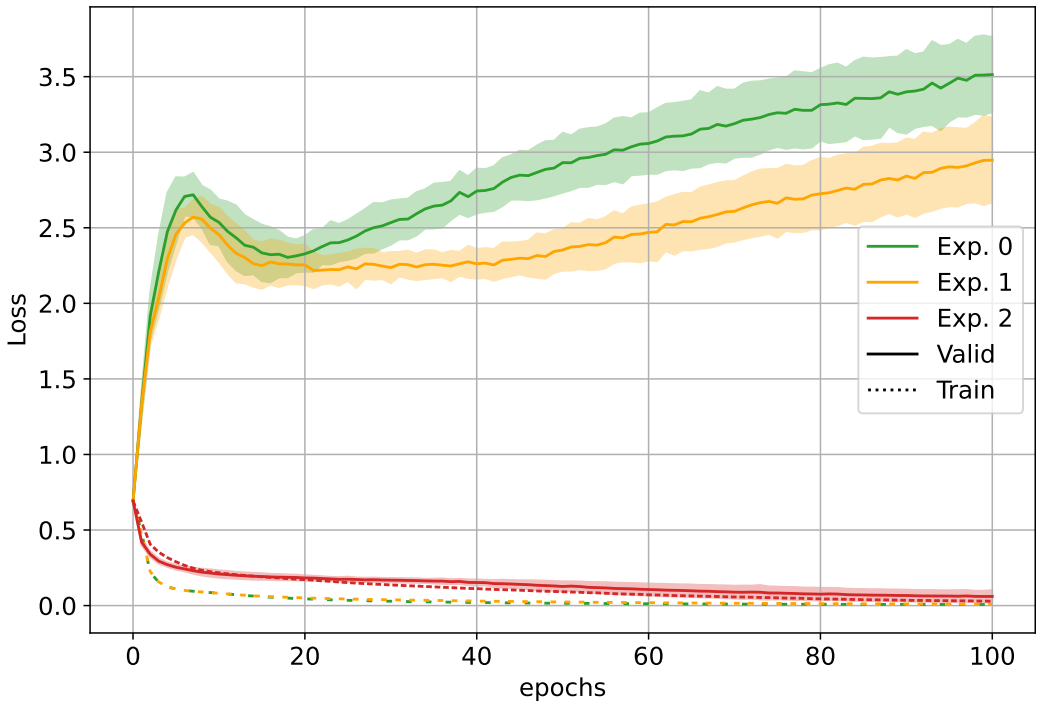


Figure 2.9 – Loss for the 3 experiments

However, the dynamics of Experiment 2 differed significantly, with the plain and dot red curves closely aligned, suggesting effective generalisation to the validation dataset. Notwithstanding, the independent test dataset resulted in a loss of 0.7, resembling the initial epoch, implying an important correlation issue between the training and validation datasets during the training phase.

An additional indication of correlation is in the delay between the green and orange dotted curves and the red one. In Experiment 2, the model required more time to attain a specific loss threshold due to the increased pattern complexity resulting from the correlation.

2.7 Conclusion

The experiments conducted show that image patches remain correlated even if they do not overlap. Having datasets correlated in this way is dangerous because it can lead to the validation of inaccurate models. Experiment 2 showed that even if a model seems relevant on the validation data, it may be useless on an independent test dataset.

It is therefore crucial to use the partitioning methods discussed in this chapter to guarantee dataset independence. Among them, the IFPO method includes and guarantees to some extent target proportions and diversity criterions. Finally, the use of geometric transformations helps to minimize the class imbalance and data lack problems.

Table 3 details the number of samples in each dataset used for the conducted training presented in the next chapter. These samples are sized 250x250x3 and do not have any overlaps. These values are provided after data augmentation.

	PD Class	**D Class**	**Total**
Train	2,936	3,360	6,296
Valid	508	656	1,164
Test	508	640	1,148
Total	3,952	4,656	8,608

Table 2.3 – Number of samples in datasets

Chapter 3: Proposed Method

To address the issue of a limited number of images, pre-trained backbones are often employed with transfer learning. However, the optimization process remains time-consuming, making it impractical to test all possible combinations of training parameters. Consequently, only the learning rate and batch size are investigated in this study, while keeping the optimizer and loss function unchanged (SGD and Binary Cross Entropy).

To begin, a preliminary step is taken to identify promising backbones that require minimal modifications to the original architecture (4.2). These selected backbones are then subjected to more time-intensive strategies for further evaluation. The first strategy involves introducing non-linearities at the end of the backbone (4.3), while the second one entails unfreezing additional layers from their original weights (4.4). Figure 3.1 provides an overview of all aforementioned procedures.

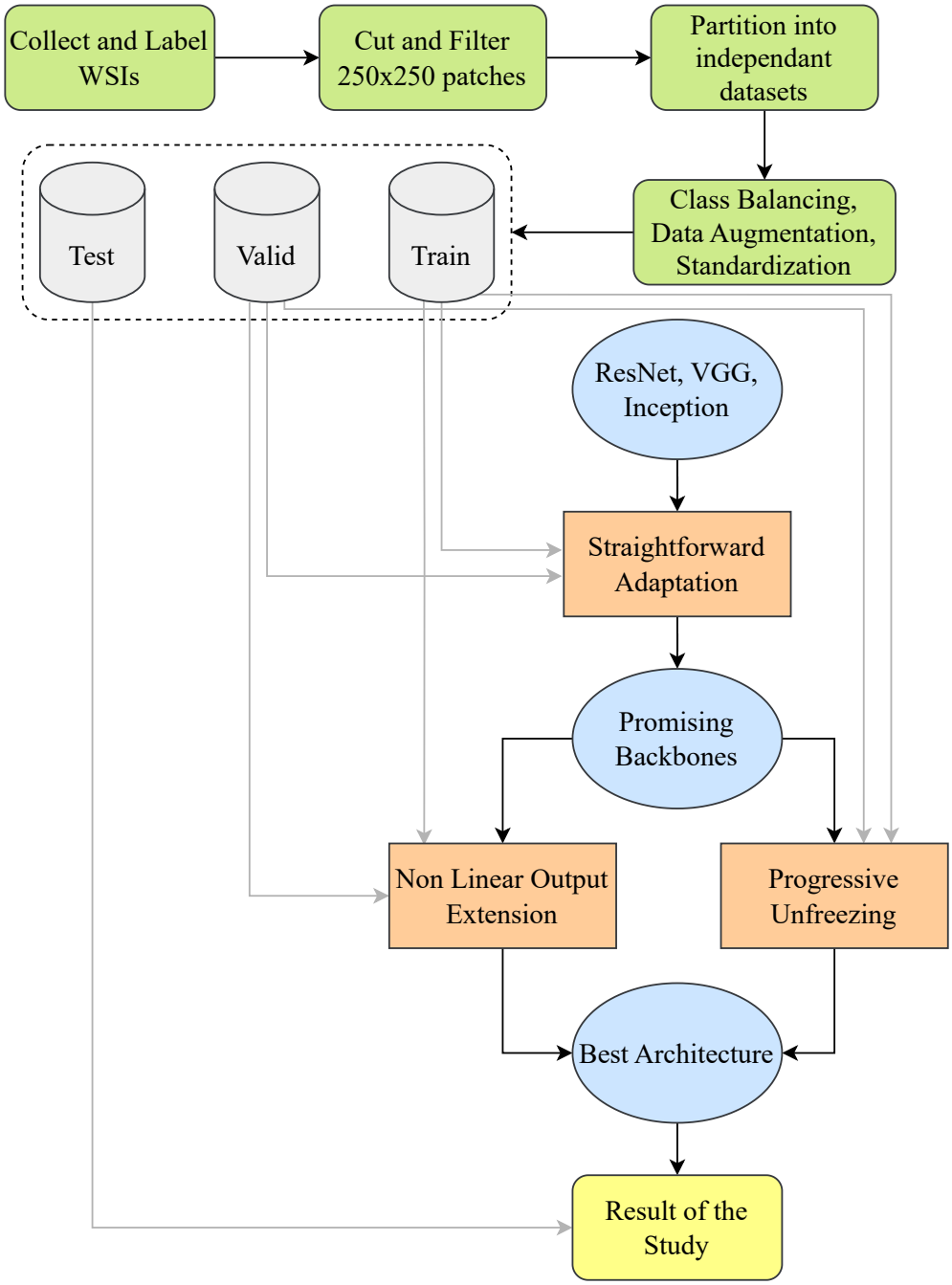


Figure 3.1 – Overview of the method

Chapter 4: Training

In this chapter, several architectures of Convolutional Neural Network (CNN) and learning approaches are tested and evaluated based on the three datasets described in chapter 2 and the method presented in chapter 3.

4.1 Metrics

4.1.1 Grid Search

During the training process, a level of randomness is present. Hence, when searching for the optimal parameter combination, such as the network version, learning rate, or batch size, several training sessions are conducted using the same values. The learning curves are presented in the appendix B as a grid search and are referenced in this chapter.

The blue curves represent the progress on the training dataset, while the orange ones refer to the validation dataset. These curves are the averages of multiple learning curves, and an opaque area is added to denotes the standard deviation. The epochs at which the best values are achieved are marked by red dots.

4.1.2 Limitations of Validation Curves

In general, the curves obtained on the training dataset are better than those on the validation dataset. The reason is that the optimisation is performed on the training one. Nevertheless, in few and rare cases, a particular initialization can lead to the curves shown in Figure 4.1. Here, the validation is better than the training on the first epochs.

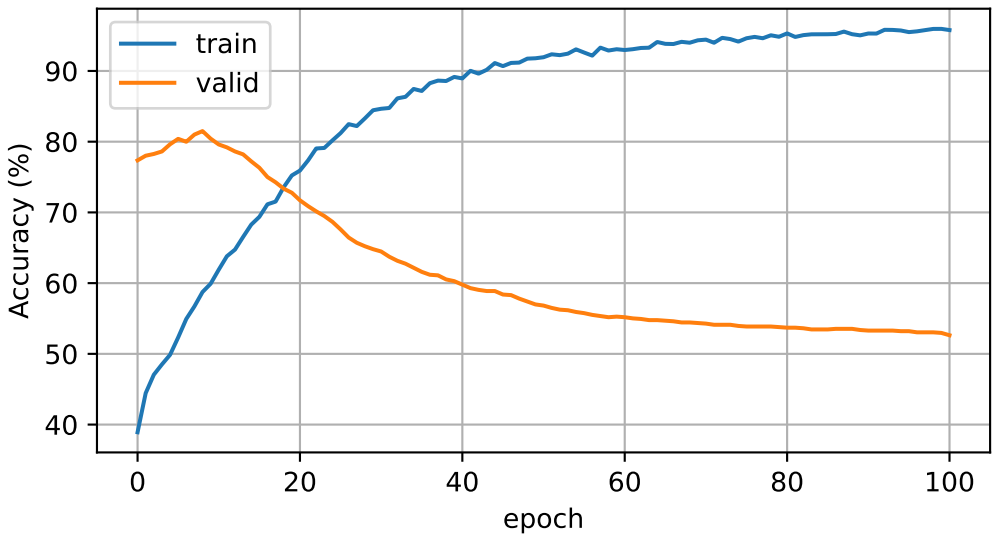


Figure 4.1 – Accuracy curves. The validation is better than the training on the first epochs

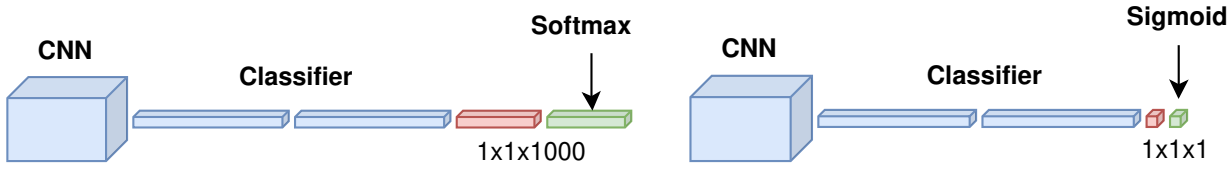
In such scenarios, solely relying on the validation curve to determine the performance of the model is inadequate. In figure 4.1, the model can not be considered to have reached more than 80% accuracy, whereas it barely reach 60% on the training at the same time. This is why the maximum of the minimum of the two curves is considered. In this particular case, this is the intersection of the two curves, just before the 20th epoch.

Similar behaviour exists on loss curves. For this metric, the minimum of the maximum of both curves is considered since the loss must be minimised.

4.2 Straightforward Adaptation

4.2.1 Idea

The straightforward adaptation involves making minimal modifications to the architecture. Specifically, the final layer must be modified to have a number of outputs that corresponds to the number of classes being predicted, except in the case of binary classification where a single output can be adequate, as illustrated by Figure 4.2. The adapted layer is subsequently trained using the specific dataset. The other layers keep their original weights.



(a) ImageNet Classifier (b) Binary Classifier

Figure 4.2 – Straightforward adaptation of the classifier

4.2.2 Results

Figure 4.3 shows the performance of the different backbones considered with a straightforward adaptation. Three families of backbones are delimited by dotted grey lines: Inception, ResNet and VGG. The optimization is performed on the loss only, and the corresponding accuracy at the same epoch is given.

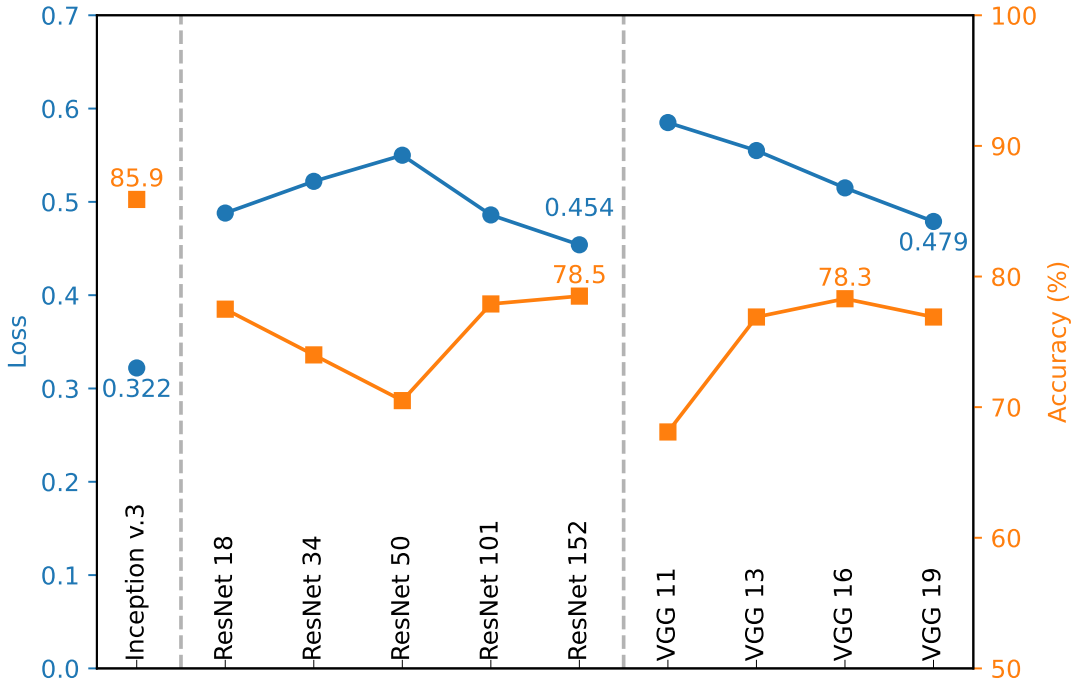


Figure 4.3 – Best loss and corresponding accuracy for the studied backbones with a straightforward adaptation

The best backbone according to both metrics is Inception3 with an accuracy of 85.9%. The corresponding grid search is provided in appendix B.1.

A second point to note from this graph is the noticeable behaviour when considering more and more sophisticated VGG architectures. As the number of convolutional layers increases, the the loss drops. However, there is no proportional or simple link between the accuracy and the loss. An illustration of it is given in appendix C.3.

The best ResNet architectures are versions 18 and 152. Unlike VGG, loss does not decrease with increasing versions. This can be explained by the variation in the classifier part, as discussed in 1.4.4. On the other hand, the loss and the accuracy curves for this family are quite symmetrical.

4.3 Non Linear Output Extension

4.3.1 Idea

Another approach is keeping the backbone with its trained weights and replacing the last layer with more than one to be trained. Between these new layers, a non-linearity function is added. This gives the network more degrees of freedom during the training phase. The added layers are fully-connected, and the number of inputs is halved from the previous one. A dropout is also inserted to prevent overfitting. Figure 4.4 illustrates this kind of classifier with 3 non-linearities in yellow, starting with 4096 inputs (as for Inception v.3).

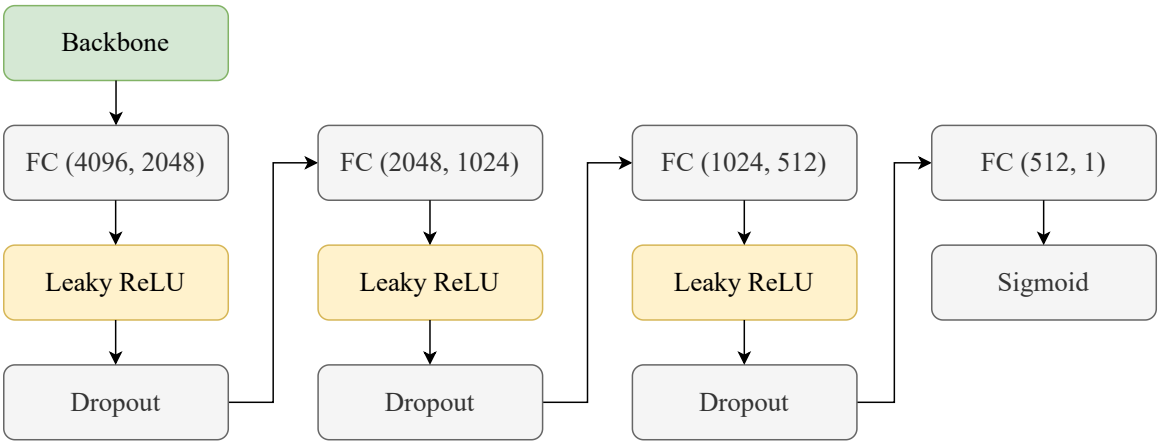


Figure 4.4 – Exemple of a non-linear output extension

4.3.2 Results

Figure 4.5 shows the results for the Inception v.3, ResNet 18, ResNet 152 and VGG 19 backbones. The loss is given as a function of the number of non-linearities introduced. The abscissa 0 corresponds to the result obtained with the straightforward adaptation with the addition of 0 non linearity (figure 4.3). Each point was obtained via a search grid, as explained in section 4.1.1.

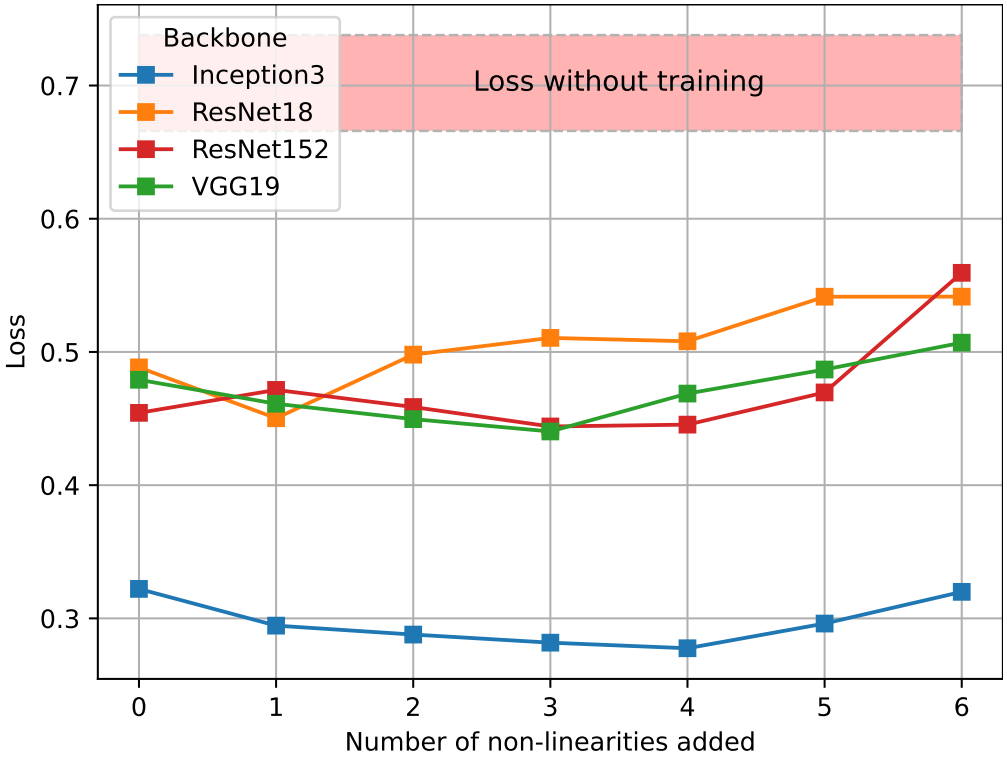


Figure 4.5 – Loss obtained with a non-linear output extension

For all tested backbones, a right-chosen extension provides an equal or a better loss than the straightforward adaptation. In particular, the Inception3 backbone with 4 non-linearities added reaches a loss of 0.278. The associated grid search is provided in annex B.2.

The red opaque region represents the loss from the training and validation datasets at epoch 0. It corresponds to the average ± standard deviation obtained from all the training sessions carried out during this work. The distribution is given in annex C.1. Within this red domain, models are akin to random decision-making.

4.4 Progressive Unfreezing

4.4.1 Idea

In the context of CNN, first layers extract general features, while last layers become more specific to the problem being studied. The straightforward adaptation approach consists in modifying only the very last layer so that it can classify according to the number of expected outputs. This layer is represented in orange in Figure 4.6.

This section develops this approach further. The weights of the last layer are still randomly initialized, and a certain amount of layers preceding the last one are unfrozen and trained from the ImageNet initialization. This idea is illustrated in red by the Figure 4.6: more and more layers are unfrozen in order to find the right tuning approach.

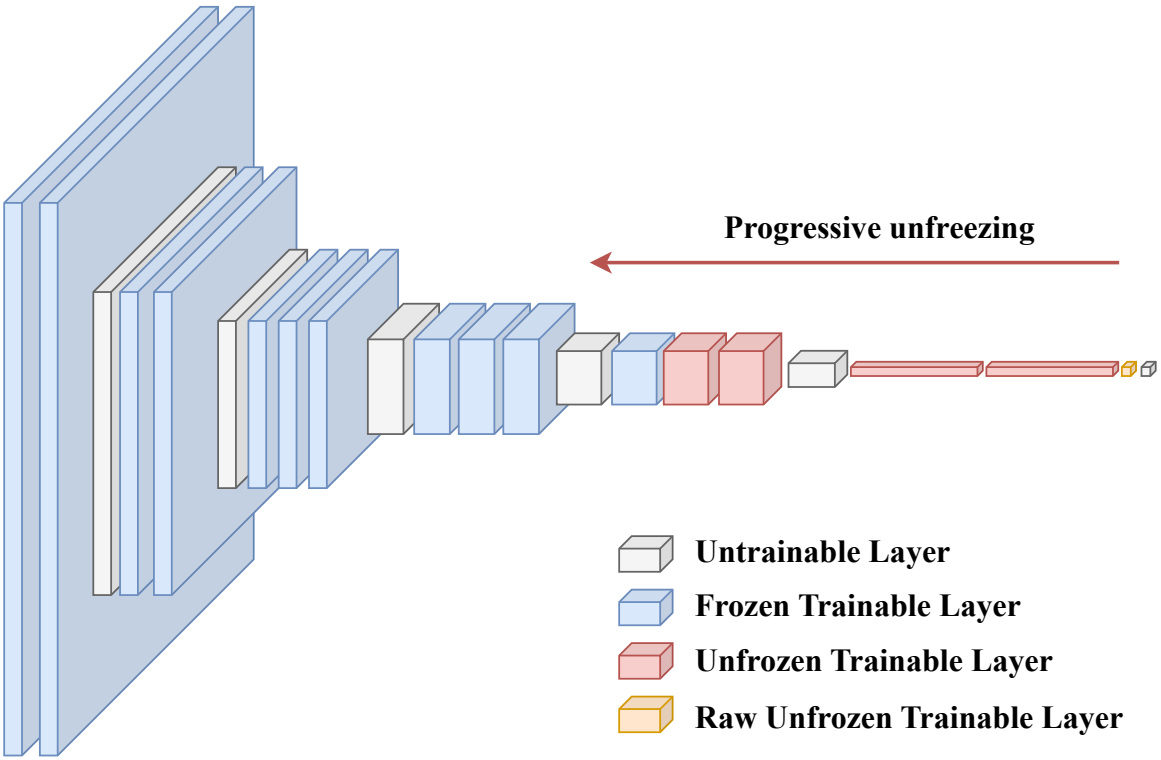


Figure 4.6 – Example of additional layers trained

4.4.2 Results

Results are given in Figure 4.7. The approach seems appropriate for VGG19. Indeed, a gain of almost 0.2 on loss is observed: this corresponds to a reduction of 40%. Progressive unfreezing is optimal here for 6 layers initialised using ImageNet weights (see Appendix B.3). Beyond that, the amount of data does not seem to allow optimisation deeper into the network.

For other networks, the approach seems less convincing: in particular for Inception v.3. This can be explained by its parallel architecture (see Appendix A.2), which requires many more layers to be unfrozen. However, there is a risk that fine-tuning will no longer be possible and the loss will increase, as is the case with VGG 19 with more than 7 unfrozen layers.

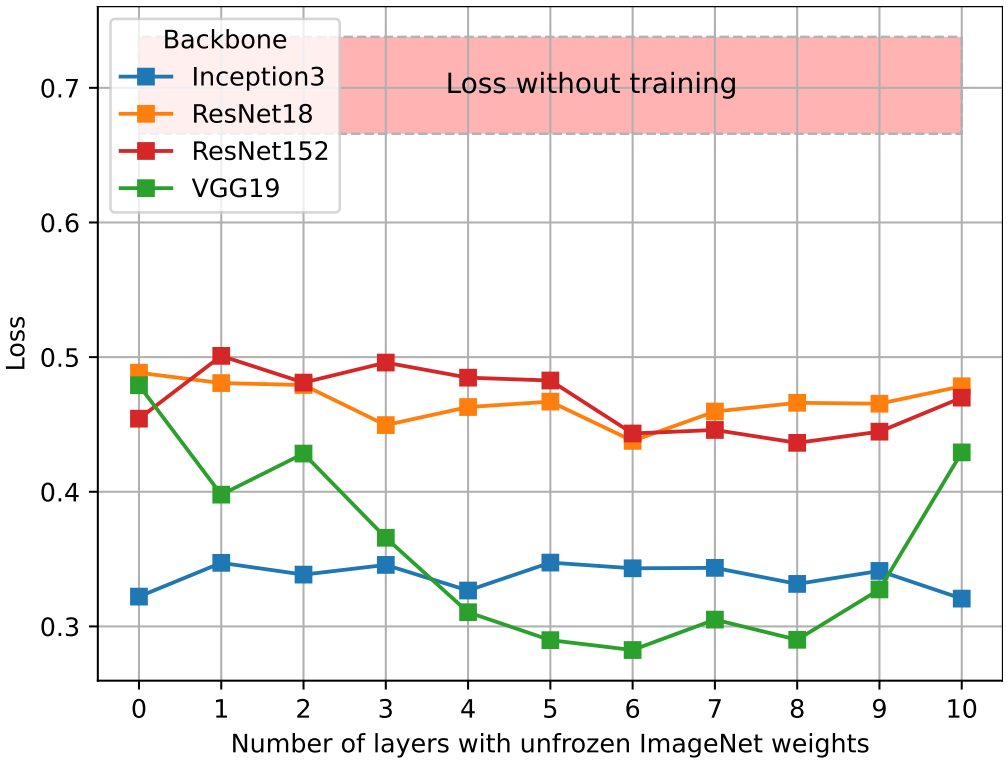


Figure 4.7 – Loss obtained with a progressive unfreezing

4.5 Conclusion

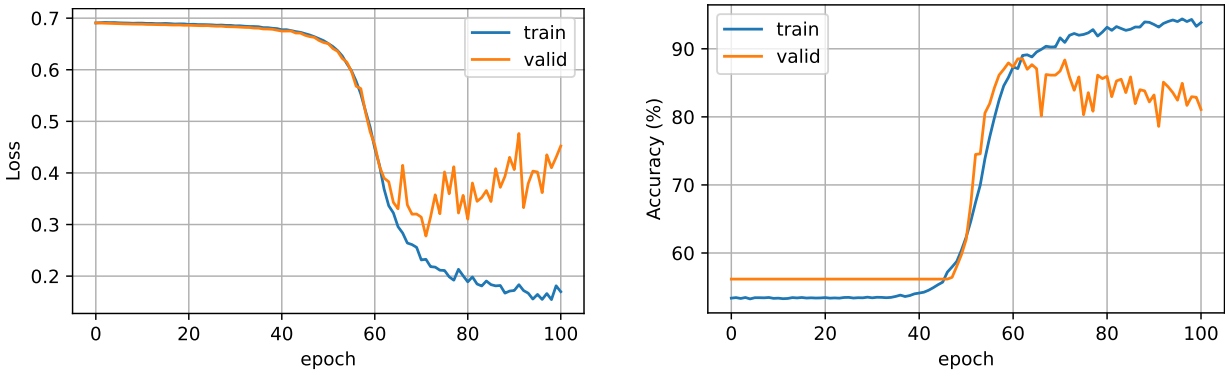
The table 4.1 summarises the best metrics obtained for the three adaptation strategies studied in this chapter. The model retained is Inception 3 with an extension of 4 non-linearities.

Experiment	CNN	Modification	Loss	Accuracy (%)
Straightforward	Inception v.3	/	0.322	85.90%
Extension	Inception v.3	4 non linearities	**0.278**	**88.36%**
Unfreezing	VGG 19	6 unfrozen layers	0.282	88.09%

Table 4.1 – Best results on validation dataset

Its training curves, which led to a loss of 0.278 on the validation dataset, are shown in Figure 4.8. The learning rate is 3×10^{-5}, the batch size 8 and the optimizer SGD. For those parameters, on the 10 training sessions performed, the mean of the best loss is 0.310 and the mean of the best accuracy 86.95%.

On the test dataset, independent of those used for training, the loss is 0.320 and the accuracy 88.43%.



(a) Loss curve

(b) Accuracy curve

Figure 4.8 – Best training session

Chapter 5: Computing Power

In this chapter, the question of the computational power required for training Convolutional Neural Networks (CNNs) is addressed. Due to limited access to sufficient resources, a solution based on Google Colaboratory has been developed.

5.1 Training Challenges and Hardware Limitations

5.1.1 Limitation of CPU Training

Training these models on a large image dataset is extremely time-consuming without adequate hardware resources. To illustrate this difficulty, consider the example of a computer equipped with an 8th generation Intel i5 processor. On this configuration, it takes more than ten hours to complete 100 epochs on the lightest backbone model in our study (ResNet-18). With the number of epochs required for grid search calculations, which typically range between 1,600 and 2,500 epochs, this approach is not achievable.

5.1.2 GPU for Computational Acceleration

A Graphics Processing Unit (GPU) is a processing unit used to accelerate graphics rendering and parallel computations. Initially, they were mainly designed to handle tasks related to image display, visual effects creation, and video game rendering. Over the past decade, GPUs have become much more powerful and versatile. They are widely used for computationally intensive tasks such as machine learning on CNNs. However, due to their cost, not all laptops have the latest generation GPUs that can efficiently perform these computations.

5.2 Google Colaboratory

5.2.1 Overview of Google Colaboratory

Google Colaboratory, also known as Colab, is a Google cloud platform that provides a Jupyter Notebook-based development environment. Colab allows users to write, execute, and share Python code directly from a browser. To some extent, this service provides free access to GPUs to accelerate computations [29].

5.2.2 Free GPU Access and Usage Limits

Google does not provide explicit usage limits for users. However, having used this service for more than 4 months, certain limits can be inferred. A user can expect to use a GPU for free for 6 hours, after which they must wait for a certain period. After this period, they can expect 4 hours of usage. The duration of usage decreases based on the history of the user but still seems to be more than 2 hours. In general, Colab prioritises active users. This is manifested through the appearance of Recaptcha tests, which, if not validated within a few minutes, revokes GPU access. These tests appear approximately after 30 minutes if the user is inactive on the browser while utilising a GPU resource.

5.2.3 Tricks to Optimise GPU Usage and Workaround Limitations

To continuously run a script, one trick is to use multiple Gmail emails. Once an account reaches the usage limit, it can be replaced. With 6 accounts, a rotation can be maintained: once the last account expires, the first one becomes usable again. After a disconnection, all variables and model states are erased. To resume training from the last checkpoint, all variables need to be updated in a Google Drive folder at the end of each epoch. Since this folder can be shared among all accounts, training can be resumed by any Google account associated with the project. Finally, a trick to avoid Recaptcha tests involves disconnecting and reconnecting to the internet once the script is launched. This changes the execution status to *busy*, preventing the appearance of tests.

5.2.4 CO2 Emission Related to Experiments

Experiments were conducted on servers based in the Western Europe, which has a carbon efficiency of 0.57 $kgCO_2eq/kWh$. A cumulative of 12,740 hours of computation was performed on hardware of type T4 (TDP of 70W). Based on those informations, total emissions are estimated to be 508.33 kg CO_2 equivalent according to the MachineLearning Impact calculator [30].

That is around half a tonne of CO2 equivalent. To put this into perspective, each French person emitted 8.9 tonnes of CO2 in 2021, according to the French Ministry for Ecological Transition [31].

5.3 Training Supervision and Management

During this project, about a hundred Gmail emails were created, enabling the execution of 18 training sessions simultaneously over the last month of this work. Throughout the entire study, 4,000 account switches were performed. Without appropriate assistance tools, managing such tasks becomes very challenging.

5.3.1 OAuth 2.0 for Access to Training Information

As mentioned earlier, intermediate results of each epoch are saved online in a shared Google Drive accessible to all used accounts. The first step in developing a supervision tool is to access this information. The chosen solution is OAuth 2.0 [32], which allows Google to share information with a third-party application without disclosing login information.

5.3.2 Dashboard Interface for Real-time Updates

Figure 5.1 is a screenshot of the developed interface. It provides real-time information with a latency of less than 1 minute about ongoing training sessions (cells marked with an orange circle). Here, we aim to perform 10 training sessions for each learning rate and batch size combination. The number of epochs performed varies between 25 and 100 depending on the batch size, the learning rate and the backbone. The dashboard also displays the next Google accounts to be used and allows visualisation of the grid search results.

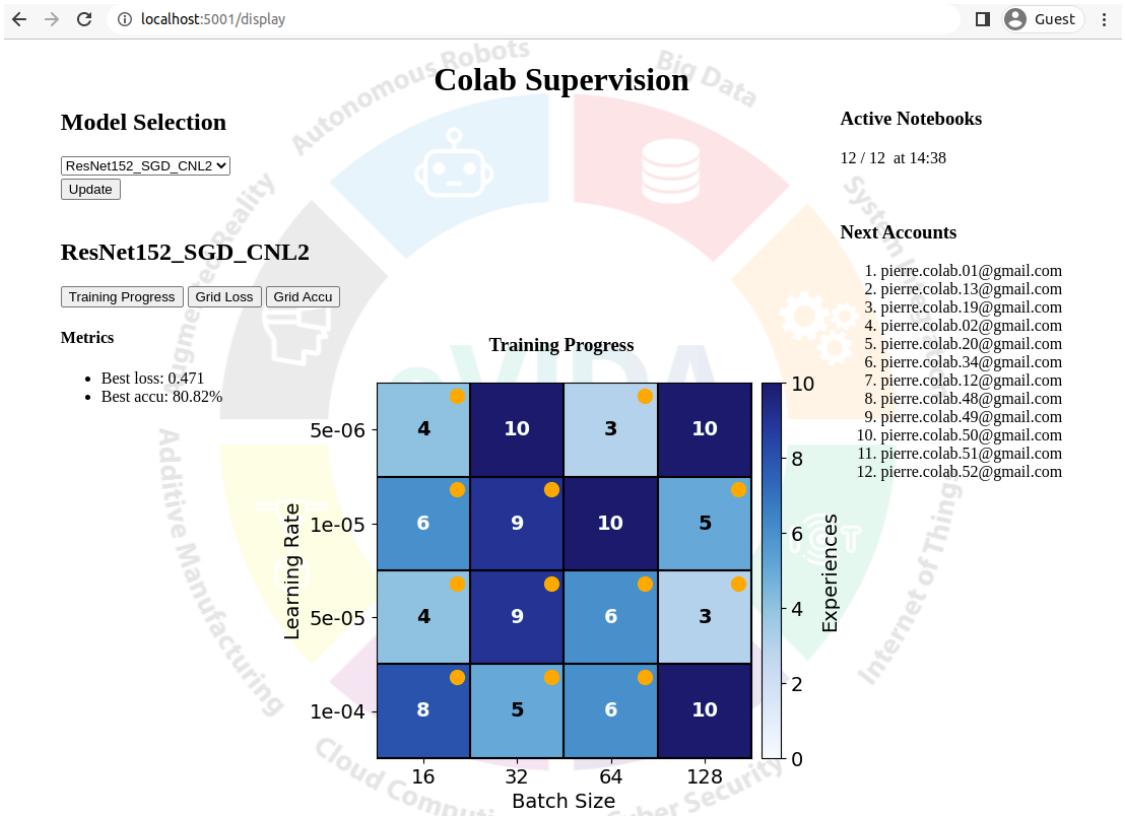


Figure 5.1 – Screenshot of the supervision dashboard used

Conclusion

Main results

The state of the art proposes solutions for classifying neuroblastoma images with scores ranging from 84% to 90%. The present study reaches this level with a score of 88.43% for binary classification.

The approach studied uses Convolutional Neural Network (CNN), and requires datasets to train, supervise and test models. These datasets are made from a database with certain characteristics that require particular attention: variable image dimensions, and a low number of images for Deep Learning applications. This led to the development of a partitioning algorithm that takes into account target proportion and diversity criteria, before increasing the data. These constraints are in addition to respecting dataset independence. The underlying question behind this independence criterion is as follows: *Can non-overlapping patches from the same image distort the interpretation of standard metrics such as the accuracy or the loss function ?* A simple experiment was set up, and the conclusion is clear: these patches are correlated, and therefore lead to misinterpretations. So patches should be assigned to the same dataset (training, validation or test) and not mixed.

The selected model is based on the architecture of Inception v.3, but it is worth mentioning that VGG 19 also yields interesting results. Both models have demonstrated good performance, albeit employing different strategies. The model based on Inception v.3 is refined by adding non-linearities at the top of the network while freezing the entire backbone with the pre-trained weights from ImageNet. On the other hand, the most relevant approach for VGG 19 is to retain the original architecture while fine-tuning not only the last layer but all classifier layers and four layers of the convolutional part.

Limitations and Future Work

It should be noted that this study is not exhaustive, as it only explores three major families of Convolutional Neural Network (CNN), while other families such as EfficientNet or DenseNet exist. Additionally, the classification performed in this study is binary, whereas the Schimada system recommends three classes: Poorly Differentiated (PD), Differentiated (D), and Undifferentiated (U) for the studied tumor. It would be beneficial to consider adding the last class U to provide a comprehensive model.

Furthermore, it is important to acknowledge that the dataset used in this study consisted of only 61 Whole Slide Images (WSIs), which were later transformed into 1,570 patches and further augmented to 8,608 samples. This database size is relatively small for training or fine-tuning certain CNN architectures that typically possess millions of parameters.

Moreover, the limitations of time and computing power pose constraints in studies of this nature, which heavily rely on powerful graphics cards. Although an approach utilizing the Google Colaboratory service was presented, it cannot be guaranteed that these methods will remain sustainable and effective in the long term.

References

[1] National Cancer Institute. *Neuroblastoma Treatment (PDQ) - Health Professional Version*. https://www.cancer.gov/types/neuroblastoma/patient/neuroblastoma-treatment-pdq. (Visited on 09/03/2023).

[2] Hiroyuki Shimada et al. « The International Neuroblastoma Pathology Classification (the Shimada system) ». In: *Cancer* 86.2 (1999), pp. 364–372. DOI: https://doi.org/10.1002/(SICI)1097-0142(19990715)86:2<364::AID-CNCR21>3.0.CO;2-7.

[3] Xintong Li et al. « A comprehensive review of computer-aided whole-slide image analysis: from datasets to feature extraction, segmentation, classification and detection approaches ». In: *Artificial Intelligence Review* 55 (Jan. 2022). DOI: 10.1007/s10462-021-10121-0.

[4] J. Kong et al. « Computer-aided evaluation of neuroblastoma on whole-slide histology images: Classifying grade of neuroblastic differentiation ». In: *Pattern Recognition* 42.6 (Oct. 2008). Digital Image Processing and Pattern Recognition Techniques for the Detection of Cancer, pp. 1080–1092. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2008.10.035.

[5] Soheila Gheisari et al. « Patched Completed Local Binary Pattern is an Effective Method for Neuroblastoma Histological Image Classification ». In: Apr. 2018, pp. 57–71. ISBN: 978-981-13-0291-6. DOI: 10.1007/978-981-13-0292-3_4.

[6] Soheila Gheisari et al. « Convolutional Deep Belief Network with Feature Encoding for Classification of Neuroblastoma Histological Images ». In: *Journal of Pathology Informatics* 9.1 (May 2018), p. 17. ISSN: 2153-3539. DOI: https://doi.org/10.4103/jpi.jpi_73_17.

[7] Panta Adhish et al. « Classification of Neuroblastoma Histopathological Images Using Machine Learning ». In: *Neural Information Processing*. Cham: Springer International Publishing, Nov. 2020, pp. 3–14. ISBN: 978-3-030-63836-8.

[8] Mohammad Norouzi, Mani Ranjbar, and Greg Mori. « Stacks of convolutional Restricted Boltzmann Machines for shift-invariant feature learning ». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 2735–2742. DOI: 10.1109/CVPR.2009.5206577.

[9] Yanfei Liu et al. « Pathological prognosis classification of patients with neuroblastoma using computational pathology analysis ». In: *Computers in Biology and Medicine* 149 (2022), p. 105980.

[10] Neha Bhardwaj et al. « Mitosis-Karyorrhexis Index evaluation by digital image visual analysis for application of International Neuroblastoma Pathology Classification in FNA biopsy ». In: *Cancer Cytopathology* 130.2 (2022), pp. 128–135.

[11] Irene Donato et al. « Demystifying neuroblastoma malignancy through fractal dimension, entropy, and lacunarity ». In: *Tumori Journal* (Jan. 2023), p. 030089162211462. DOI: 10.1177/03008916221146208.

[12] Justin M. Johnson and Taghi M. Khoshgoftaar. « Survey on deep learning with class imbalance ». In: *Journal of Big Data* 6.27 (Jan. 2019). DOI: https://doi.org/10.1186/s40537-019-0192-5.

[13] Mingle Xu et al. *A Comprehensive Survey of Image Augmentation Techniques for Deep Learning*. Nov. 2022. DOI: 10.48550/ARXIV.2205.01491.

[14] Zewen Li et al. « A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects ». In: *IEEE Transactions on Neural Networks and Learning Systems* 33.12 (Dec. 2022), pp. 6999–7019. DOI: 10.1109/TNNLS.2021.3084827.

[15] Shaofeng Cai et al. « Effective and Efficient Dropout for Deep Convolutional Neural Networks ». In: *CoRR* abs/1904.03392 (July 2020). arXiv: 1904.03392. URL: http://arxiv.org/abs/1904.03392.

[16] Karen Simonyan and Andrew Zisserman. « Very Deep Convolutional Networks for Large-Scale Image Recognition ». In: (2015).

[17] Shaoqing Ren Kaiming He Xiangyu Zhang and Jian Sun. « Deep Residual Learning for Image Recognition ». In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[18] Christian Szegedy et al. « Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning ». In: (2016). arXiv: 1602.07261 [cs.CV].

[19] *PyTorch - Models and pre-trained weights*. https://pytorch.org/vision/stable/models.html#table-of-all-avai 2023.

[20] Mingxing Tan and Quoc V. Le. « EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks ». In: *CoRR* abs/1905.11946 (Sept. 2020). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[21] Lorenzo Ciampiconi et al. *A survey and taxonomy of loss functions in machine learning.* Jan. 2023. DOI: 10.48550/ARXIV.2301.05579.

[22] J. Kiefer and J. Wolfowitz. « Stochastic Estimation of the Maximum of a Regression Function ». In: *Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466. DOI: 10.1214/aoms/1177729392.

[23] Papers with Code. *SGD with Momentum.* Accessed on March 23, 2023. Papers with Code. 2023. URL: https://paperswithcode.com/method/sgd-with-momentum.

[24] Singer Duchi Hazan. « Adaptive subgradient methods for online learning and stochastic optimization ». In: *Journal of Machine Learning Research* 12 (2011). Accessed on March 23, 2023, pp. 2121–2159. URL: https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf.

[25] Tijmen Tieleman. *CSC321: Neural Networks and Machine Learning, Lecture 6: Optimization.* Accessed on March 23, 2023. University of Toronto. 2015. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[26] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2017. arXiv: 1412.6980 [cs.LG].

[27] Chiagoziem C. Ukwuoma et al. « Multi-Classification of Breast Cancer Lesions in Histopathological Images Using DEEP_Pachi: Multiple Self-Attention Head ». In: *Diagnostics* 12.5 (May 2022). ISSN: 2075-4418. DOI: 10.3390/diagnostics12051152.

[28] Fangfang Gou et al. « A Multimodal Auxiliary Classification System for Osteosarcoma Histopathological Images Based on Deep Active Learning ». In: *Healthcare* 10.11 (Oct. 2022). ISSN: 2227-9032. DOI: 10.3390/healthcare10112189.

[29] *Google Colaboratory.* URL: https://colab.research.google.com/ (visited on 06/04/2023).

[30] Alexandre Lacoste et al. *Quantifying the Carbon Emissions of Machine Learning.* 2019. arXiv: 1910.09700 [cs.CY].

[31] French Ministry for Ecological Transition. *L'empreinte carbone de la France de 1995-2021.* URL: https://www.statistiques.developpement-durable.gouv.fr/lempreinte-carbone-de-la-france-de-1995-2021 (visited on 06/16/2023).

[32] Auth0. *What is OAuth 2.0?* Auth0 website. URL: https://auth0.com/intro-to-iam/what-is-oauth-2 (visited on 05/29/2023).

Appendix A: CNN Architecture

I Visual Geometry Group

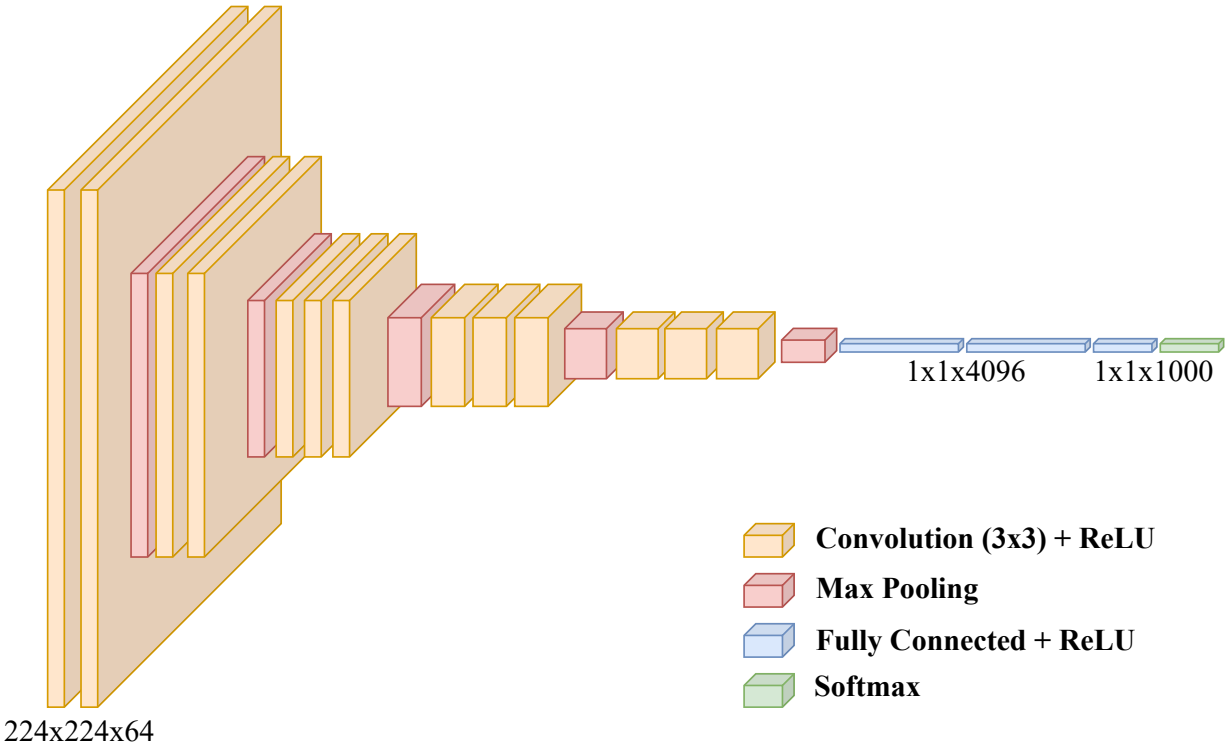


Figure A.1 – VGG 16

II Inception v.3

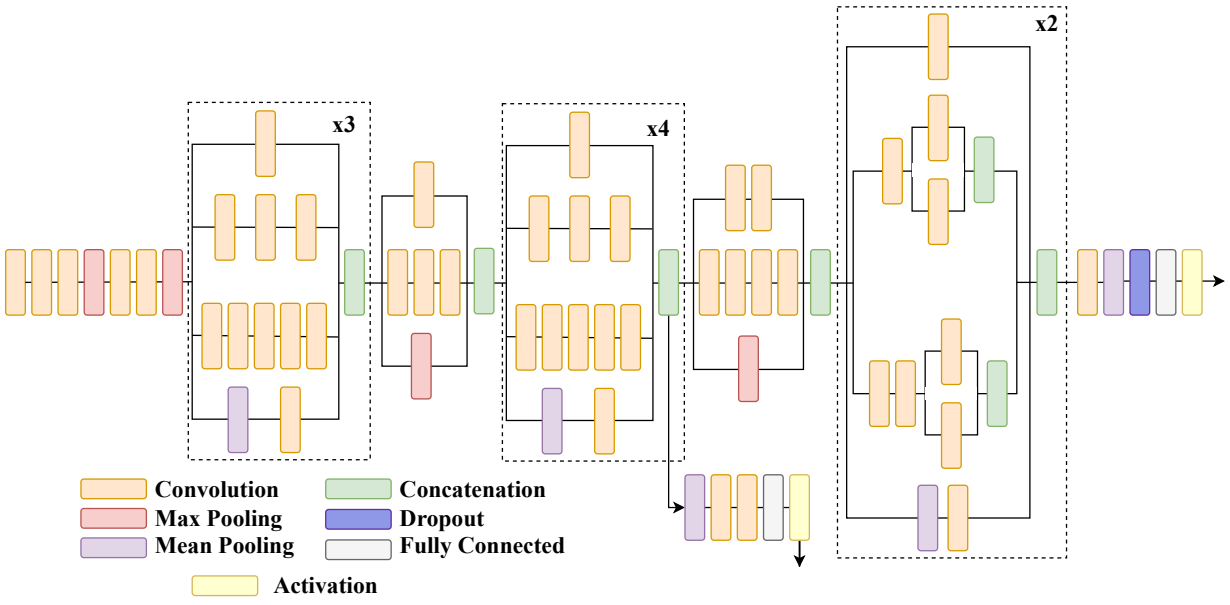


Figure A.2 – Inception v.3

Appendix B: Grid Search

I Straightforward Adaptation

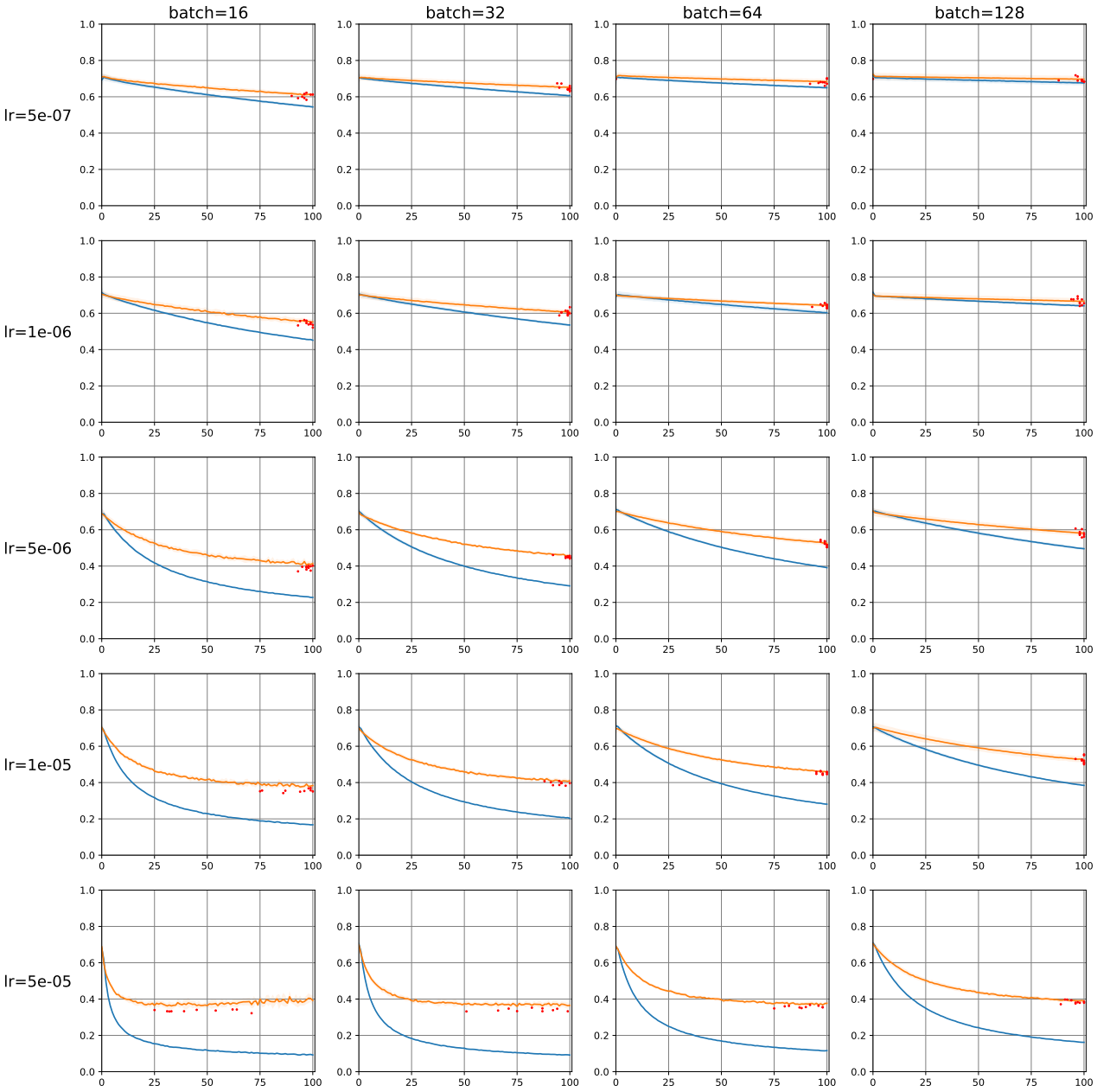


Figure B.1 – Loss for Inception3 optimized using SGD Momentum. Exhaustive search for learning rate and batch size. Each pair is tested ten times. Best: 0.322

II Non Linear Output Extension

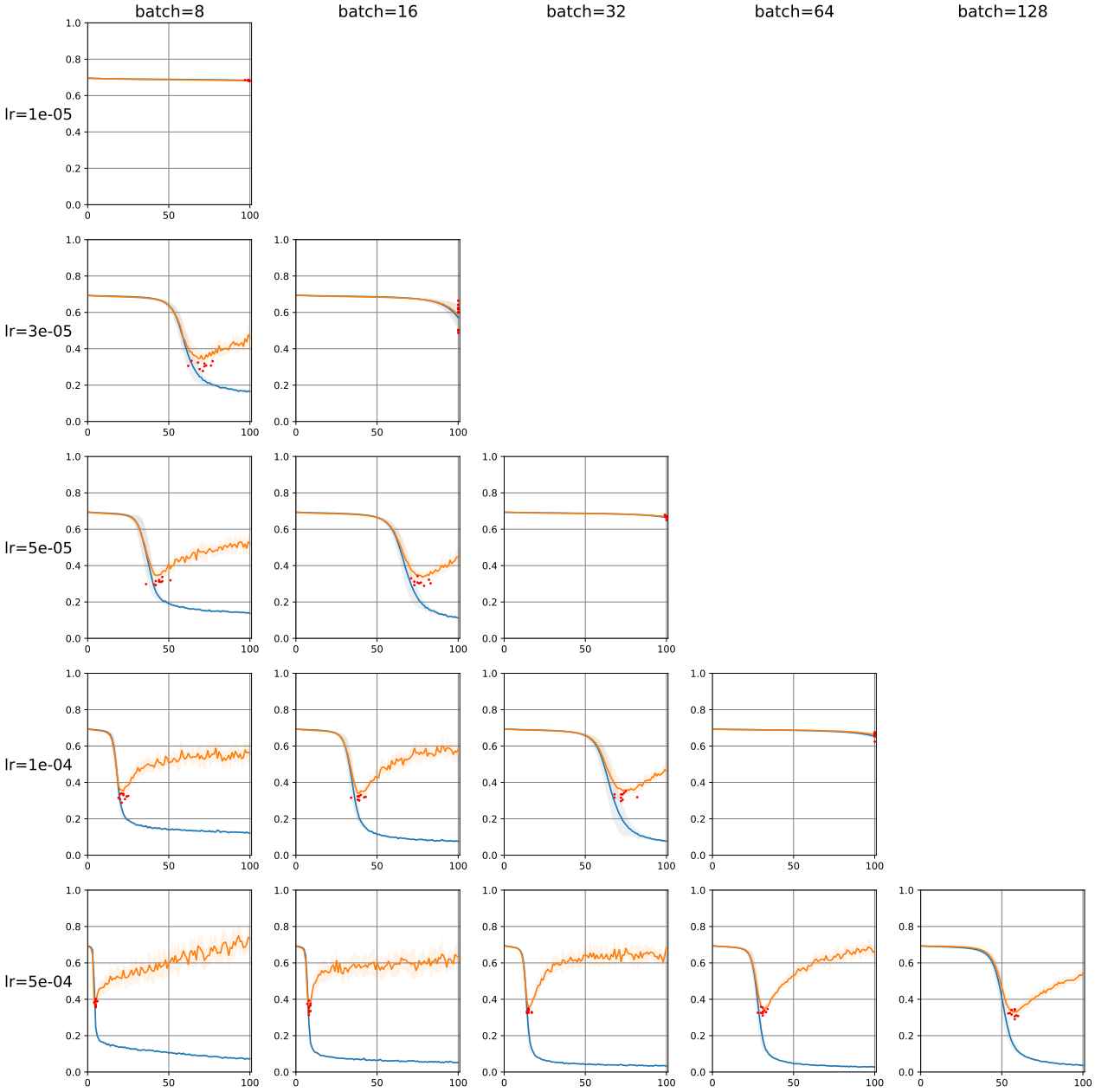


Figure B.2 – Loss for Inception3 with 4 non-linearities added and optimized using SGD Momentum. Exhaustive search for learning rate and batch size. Each pair is tested ten times. Best: 0.278

III Progressive Unfreezing

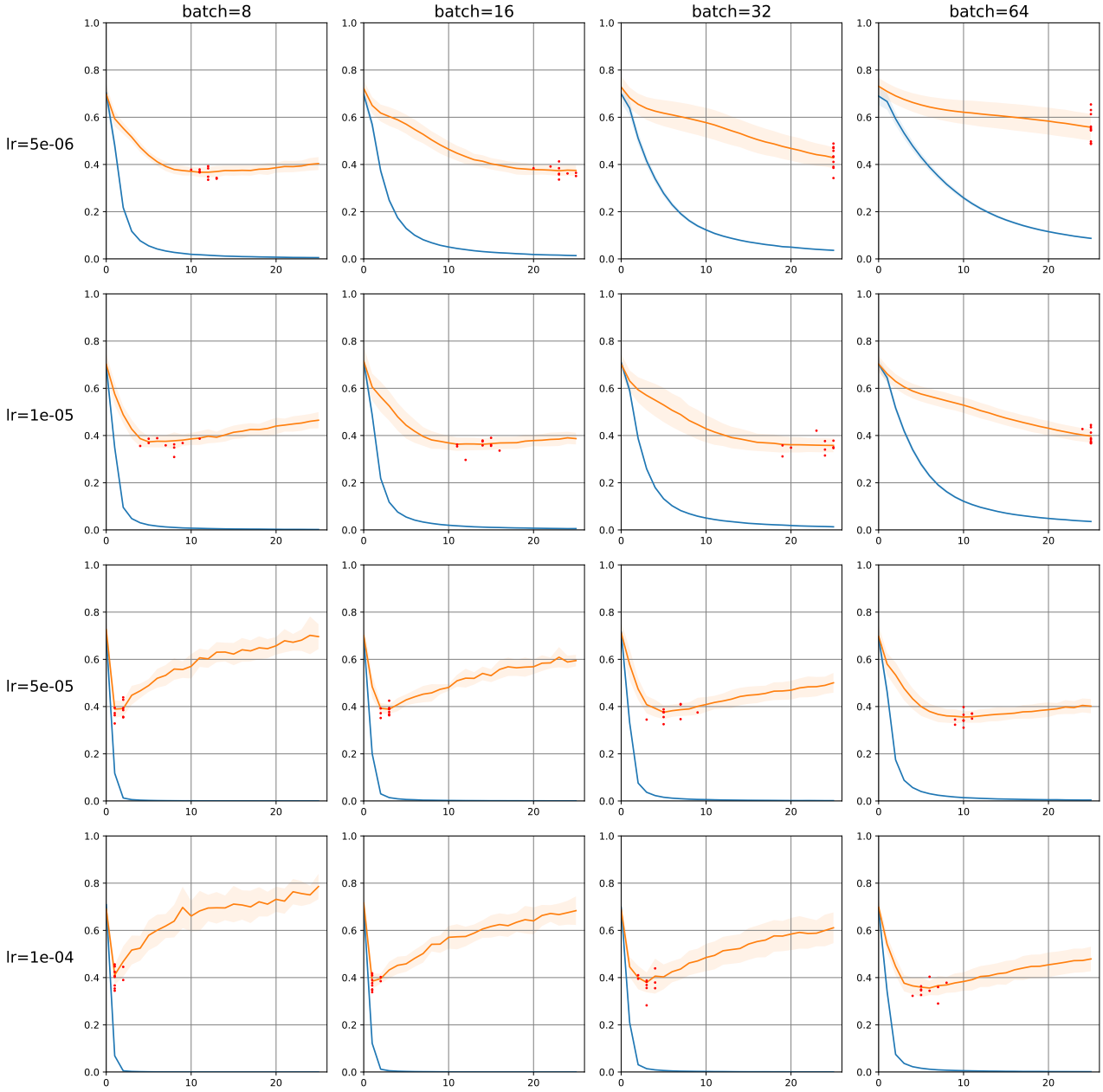


Figure B.3 – Loss for VGG19 with 6 unfrozen layers initialized with ImageNet weight and optimized using SGD Momentum. Exhaustive search for learning rate and batch size. Each pair is tested ten times. Best: 0.282

Appendix C: Distribution

I Loss

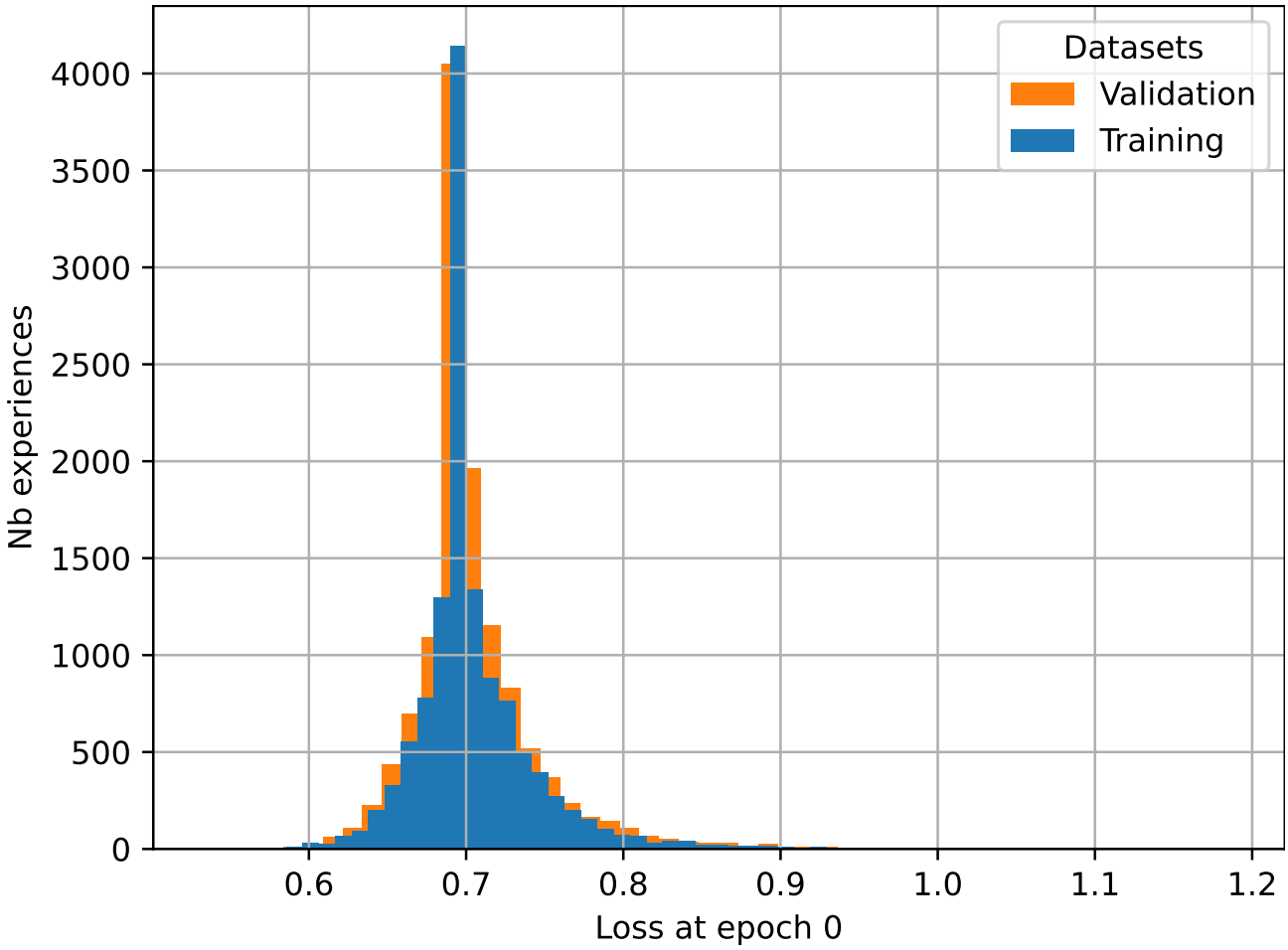


Figure C.1 – Loss distribution at epoch without training from 12,500 sessions. Mean: 0.705, std: 0.042

Without few outliers after 0.85, this loss distribution seems to be Gaussian. This is why the loss without training is represented by an area of mean ± std in Figures 4.5 and 4.7.

II Accuracy

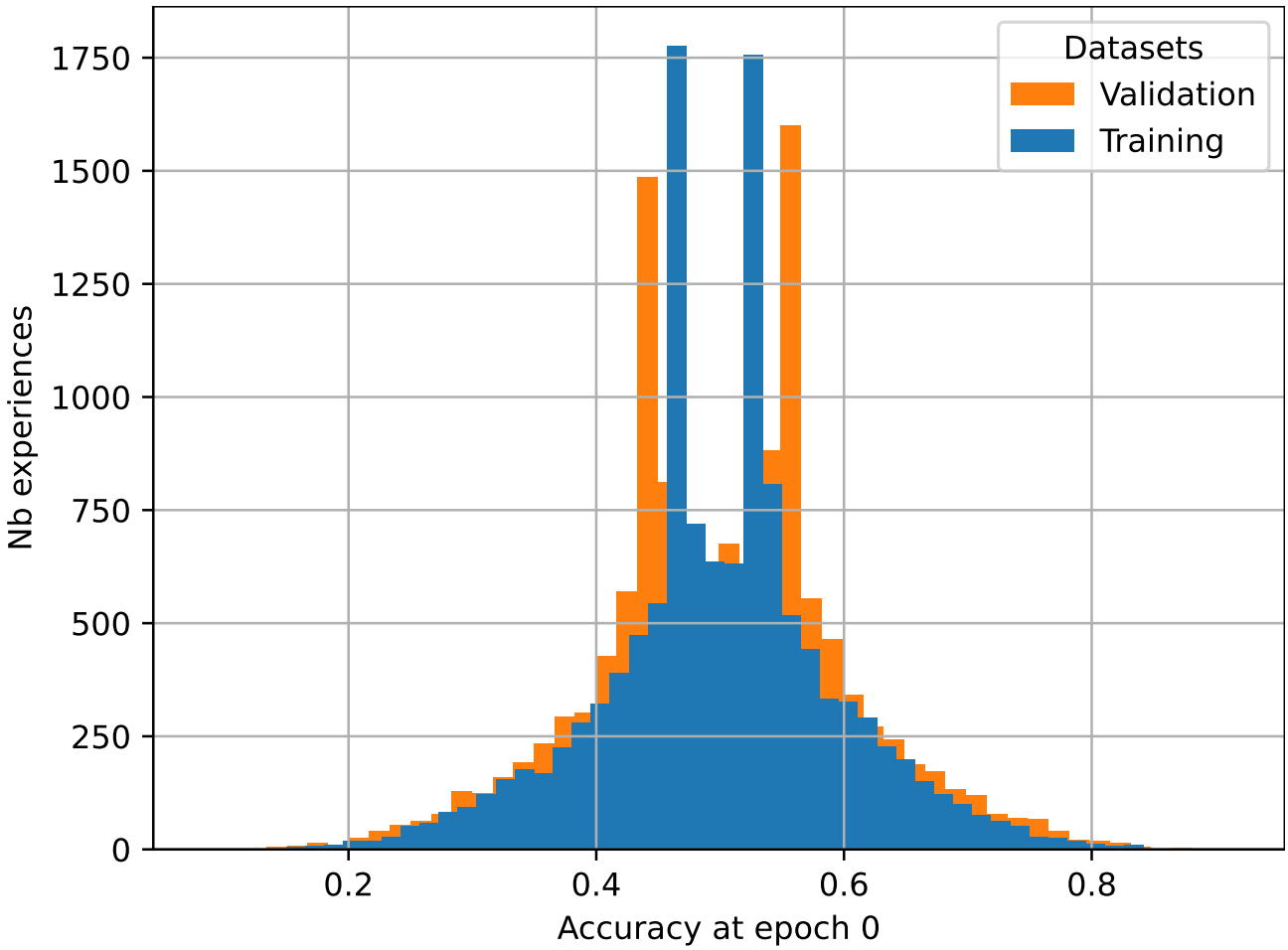


Figure C.2 – Loss distribution at epoch without training from 12,500 sessions. Mean: 0.500, std: 0.10

The distribution according to performance differs from the one according to loss, even though these are exactly the same sessions that initiated the training sessions. This clearly shows that these two metrics are different and both deserve to be analysed.

The average obtained is 0.5, which is typical for a binary random classification. However, the distribution does not have a Gaussian shape due to the presence of two peaks around this mean. One possible explanation lies in the correlation between the patches in large images used in the different datasets. If a patch is classified in a certain way, it is likely that the rest of the patches will also be classified in the same way, resulting in an entire part of the dataset. The imbalance arises from the fact that the number of patches extracted is not the same for each WSIs. Therefore, the binary nature of the accuracy affects the distribution. However, this is not the case for the distribution according to the loss function, as the loss is continuous.

III Loss and Accuracy

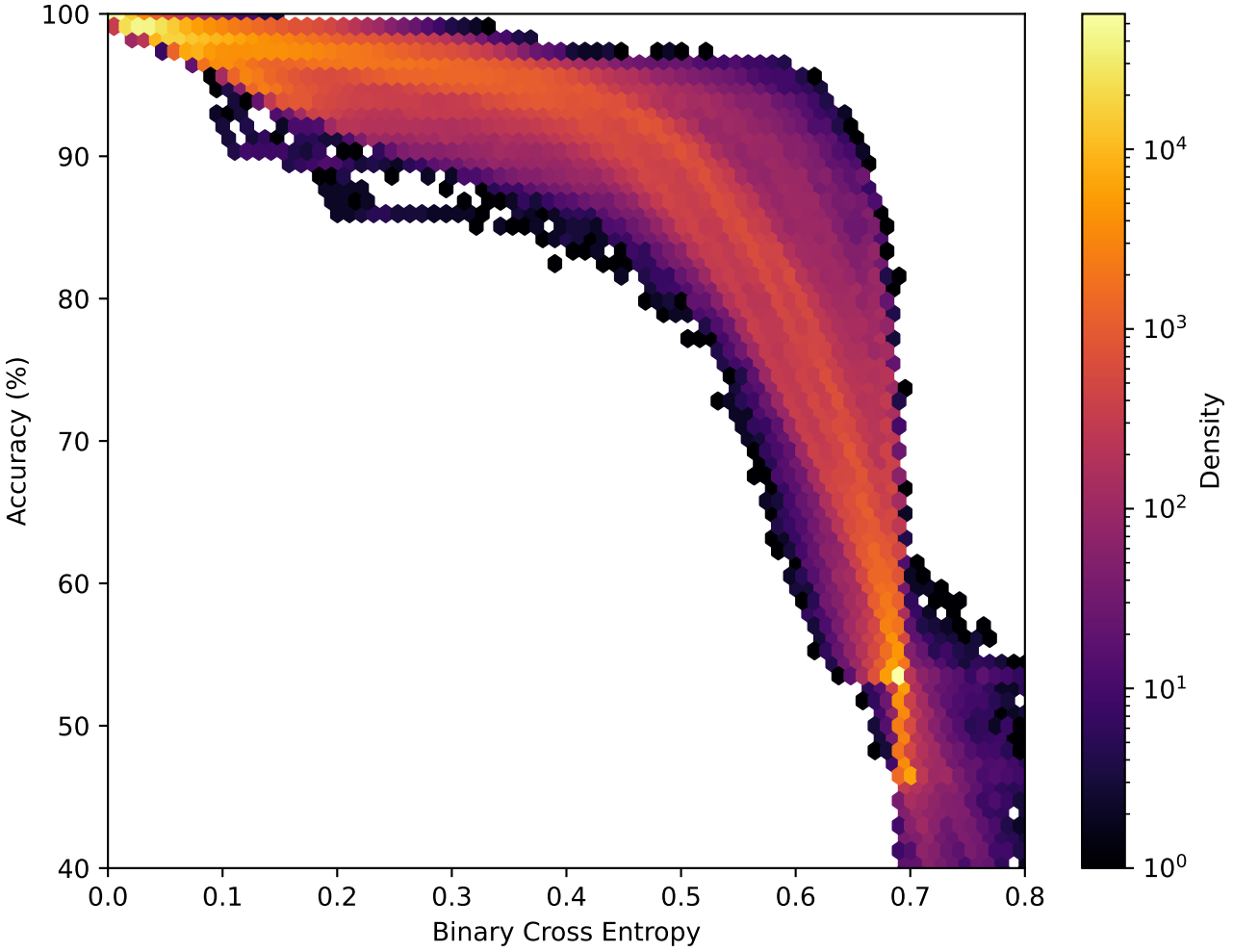


Figure C.3 – Loss and accuracy distribution over 936,000 epochs on a logarithmic scale.

Figure C.3 shows a 2D histogram describing the distribution of the pairs (accuracy, loss) for common epoch through all training session performed. A strong match is indicated by lighter colours. It can be seen that the relationship between accuracy and loss is non-linear and quite diffuse (represented by purple areas), although a trend can be discerned in yellow-orange.