FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Understanding Regularization in Deep Neural Networks: a Metalearning Approach

Miguel Mano

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Carlos Soares

Second Supervisor: Pedro Abreu

July 21, 2020

Understanding Regularization in Deep Neural Networks: a Metalearning Approach

Miguel Mano

Mestrado Integrado em Engenharia Informática e Computação

July 21, 2020

Abstract

In recent years, deep learning has become more and more useful, solving increasingly complicated applications with greater accuracy over time, particularly in computer vision and natural language processing. However, important technical questions are raised such as high computational costs and their proneness to overfit. The latter is defined as an indicator of a network's generalization capabilities, *i.e.*, its ability to perform well on previously unobserved inputs.

As an attempt to minimize overfitting, several regularization techniques – which are usually explicitly designed to reduce the test error at the expense of increased training error – are widely available, such as early stopping, weight decay and layer dropout. Despite their widespread use and known capabilities, literature provides very little insight on how to apply said strategies or guidelines on how much they positively (or negatively) impact a network's overfitting issue. The idea of experimenting to see what works and the inability to identify just why it does further solidifies the black-box design neural networks are known for and that has slowed implementation of deep systems on industries that need explainable networks.

This dissertation is a first attempt at modeling dropout – a highly performant and immensely popular regularization technique – with meta-learning. Essentially summarized as "learning to learn", meta-learning is the use of ML techniques to infer their own behaviour. It is typically used for hyperparameter tuning and algorithm selection but its capability of generalizing to environments never encountered before also unlocks considerable insight towards a network's thought process, *per se*. Meta-learning requires the extraction of dataset characteristics – dubbed "meta-features" – which can be extracted in many ways.

The approach dives into the domain of image classification and, therefore, makes use of Convolutional Neural Networks (CNN). This architectural choice raises the challenge of developing appropriate data characteristics as existing ones are for tabular data, given that meta-learning has yet to be applied to a field like computer vision. We extensively analyse a proposal for meta-features in recent work and propose a new approach to data characterization and empirically evaluate it.

Results confirm that designing suitable meta-features for image classification is an intricate process with numerous interesting challenges in that area.

Keywords: deep learning, convolutional neural networks, meta-learning, regularization, dropout

Resumo

Recentemente, *deep learning* tem-se tornado mais e mais útil, resolvendo aplicações incrementalmente complicadas com alta eficácia, particularmente nos campos de visão de computador e processamento de linguagem natural. Porém, o crescimento progressivo em tamanho e complexidade de modelos *deep learning* terá sido acompanhado de questões técnicas importantes como custos computacionais elevados e a tendência ao *overfit*. Overfit é definido como um indicador das capacidades de generalização de uma rede, isto é, a sua aptidão para ser eficaz em *inputs* previamente inobservados.

Numa tentativa de minimizar overfitting, várias técnicas de regularização – usualmente explicitamente constituídas para reduzir o erro de testes à custa de um erro de treino maior – estão disponíveis, como *early stopping*, *weight decay* e *dropout* de camadas. Ainda assim, apesar da sua popularidade e conhecida eficácia, a literatura fornece pouco conhecimento em como aplicar as estratégias para solucionar o *overfitting* ou guias sobre os seus impactos (positivos ou negativos). A ideia de experimentar para ver o que funciona e a inaptidão de identificar precisamente o porquê que funciona solidifica ainda mais o design de caixa-negra inerente a redes neuronais que tem sido o entrave à implementação de sistemas *deep* em indústrias que necessitam de redes explicáveis.

Esta dissertação tenta expandir em explanabilidade de regularização, utilizando uma abordagem de *metalearning*. Essencialmente sumarizado como "aprender a aprender", *metalearning* consiste no uso de técnicas de *machine learning* para inferir o seu próprio comportamento. É típicamente utilizado para ajustes de hiperparâmetros e seleção de algoritmos, mas é a sua capacidade de generalização para ambientes nunca antes explorados que desbloqueia conhecimento considerável sobre a forma de pensar de uma rede, por assim dizer.

A abordagem planeada contém experimentação com *dropout* – uma técnica de regularização popular e altamente eficaz – para prevenir *overfitting* e subsequentemente impulsionar a eficácia de redes neuronais convolucionais e inferir meta-modelos que representam conhecimento sobre a sua eficiência. Esta escolha arquitetural levanta o desafio de desenvolver caraterísticas de data apropriadas, visto que as existentes são para informação tabelar, dado que *metalearning* ainda não terá sido aplicado em larga escala a um campo como visão por computador.

Keywords: metalearning, regularização, deep learning, redes neuronais convolucionais

Acknowledgements

I thank my supervisors, Professors Carlos Soares and Pedro Abreu for their invaluable insights on how to elaborate a master thesis and perform proper research work. Our lighthearted and dynamic meetings made for a very enjoyable learning experience in the machine learning field.

I also thank the Fraunhofer Portugal Research Center for allowing me to use their high-performance cluster and especially João Gonçalves for restlessly aiding me in configuring everything to get the experiments running.

Finally, I thank my parents who patiently listened to me describing way too many deep learning concepts for way too much time and supported me throughout, my friends who always made sure to keep me sane and entertained, and girlfriend for always ensuring that I remained mentally motivated and encouraged me to stay fit during these COVID-19 lockdown times.

Miguel Mano

"In an interstellar burst,
I am back to save the universe."

Radiohead

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial Neural Network
AUC	Area Under the Curve
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CT	Computer Tomography
DCT	Discrete Cosine Transformation Domain
DNN	Deep Neural Network
FEUP	Faculty of Engineering of the University of Porto
FN	False Negative
FP	False Positive
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
HPC	High-performance Computing
LIACC	Artificial Intelligence and Computer Science Laboratory
MLP	Multilayer Perceptron
MR	Magnetic Ressonance
MSD	Medical Segmentation Decathlon
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristic
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

Chapter 1

Introduction

The biggest evolutionary step for deep learning happened in 1999, where computers started becoming faster at processing data and GPUs (graphics processing units) were introduced, ending the AI winter the field had had from 1985 to the 1990s. This made way for an increase in computational speeds by 1000 times over a 10 year span that allowed neural networks to compete with SVMs (support vector machines), one of the most popular predictors at the time [17]. The expression *deep learning* gained popularity to illustrate how researchers were now able to train deeper neural networks than even before [43].

Due to the vast array of network types that specialize in specific tasks, deep learning is capable of being a viable and excellent solution for a myriad of problems. For instance, Recurrent Neural Networks (RNNs) – networks where data can flow in any direction – have been proven highly competent on large scale language modeling, fundamental for language understanding [27]. Generative Adversarial Networks' (GANs) ability to generate new data given a training set has been useful to recover features in astrophysical images of galaxies [42], up-scaling low-resolution video games [51] and generating or swapping human faces to an almost lifelike degree. Convolutional Neural Networks (CNNs) and their effectiveness in image recognition and classification have been indispensable for automatically generating image descriptions [28], performing style transfer [19], facial recognition and many more. Figure 1.1 shows a chart containing the most popular existing neural network architectures right now.

1.1 Context and motivation

Deep neural networks are artificial neural networks with more complex architectures. They are distinguished from basic artificial neural networks by their depth and the inclusion of multiple "hidden" layers between the input and output layers. These networks learn by adjusting the weights of

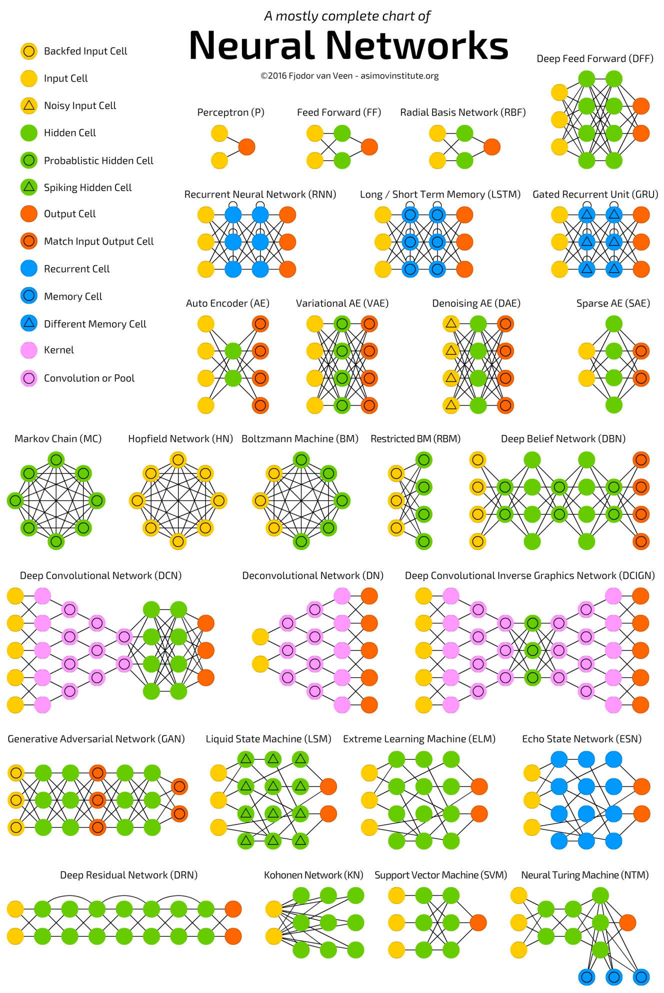

A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Figure 1.1: Examples of the most popular existing neural network architectures [5]

connections between neurons, across multiple layers. Activation functions attach to each neuron and determine whether its input is generally useful for the prediction. It is this adjustment mechanism that guides networks towards converging on an output. However, in some circumstances, models are unable to generalize beyond training data. This is known as *overfitting*.

Nowadays, the explosion of the big data field means data scientists handle an increasingly higher amount of data. This abundance comes with a curse and makes feature selection, dimension reduction and regularization steps become more and more necessary to avoid the bane of overfitting [15]. As an example, the ImageNet dataset has over 15 million labeled high-resolution images, spread across about 22,000 categories. One of the largest convolutional neural networks to date was trained to attempt it and the network's size was reported as the source of a significant overfitting issue [31].

To better understand overfitting, we introduce the concept of *capacity*: a model's ability to fit a wide variety of functions. Any model should have an appropriate level of capacity for the complexity of the task at hand and the amount of training data they are provided with [24]. When a network's capacity happens to be higher than needed, it overfits because it is training on it too much. If a model is said to have low capacity, then it is failing to find meaningful patterns on the data and is not training well enough.

While overfitting is an integral part of the deep learning process, there is a wide range of regularization techniques to eliminate it or potentially mitigate their overall impact. The most popular techniques are:

- **L_1 regularization** – also known as Lasso Regression – reduces less important variables to zero by adding the summation of the weights of the variables to the cost function. **L_2 regularization** – also known as Ridge Regression – reduces them to values close but not zero by adding the summation of squared weights of the variables to the cost function. Overall, these techniques limit the capacity of models by adding a parameter norm penalty to the objective function;
- **Dropout** lets a model train and evaluate an ensemble of networks by removing non-output units. Simply put, a set of neurons are deactivated at random during the training phase with set probability, removing all incoming and outgoing edges to the units in the set. While training, neurons naturally develop co-dependency that hinders their individual power. The theory behind it is that powering down neurons at random ensures outputs are not as dependent on eachother, preventing overfitting;
- **Early stopping** halts all training as soon as validation error increases. In other words, it only trains up until the point where the model starts overfitting.

These strategies are valid for various contexts and useful in specific situations. However, a big portion of development time still boils down to experimenting with which technique yields the best results on any given situation. Nowadays, data scientists that perform any type of machine learning eventually adopt a trial-and-error methodology at some stage of development. There is a vast set of algorithms and network architectures available for usage on many different contexts and literature is at times unclear as to when to use said technique and its expected outcomes. Additionally, artificial neural networks expose an overwhelming number of hyperparameters to tune a model. Significant strides towards solving the hyperparameter optimization problem have been made, like more sophisticated implementations of grid searching – in its core, a bruteforce way of testing and evaluating a set of hyperparameter combinations. Nonetheless, the performance of many machine learning methods still depend on manually engineered features and hyperparameter settings and are the cause of either mediocre or state-of-the-art results [23].

This is partially why, as of late, the scientific community's efforts have turned towards the explainability of neural networks. A disadvantage of deep neural networks is their internal logic of abstract nature and of difficult interpretation, a characteristic known as a *"black box"* design [22]. Breaking deep learning's black box is an asset to both saving development time and reassuring industries where artificial intelligence choices are critical and must be made clear for human validation, such as the medical or the banking industries.

When presented with an overfitting scenario, the very same trail-and-error approach reoccurs. Although the literature is rich in regularization technique variety, it does not specifically explain when to pick one over another and its expected effects. At the very least, data scientists should have a rough guideline on which techniques are fit for any given context, in an attempt to reduce development time.

1.2 Objectives

The dissertation's goal is to study relationships between a network's input, the selected regularization technique and the model's performance. We aim to advance the literature regarding the explanability of regularization techniques and their optimal use cases on convolutional neural networks. During this path, it is expected to achieve advancements as to which approaches for extracting metafeatures from images are proven effective. This is a surprisingly challenging task due to the lack of literature on the subject. For this purpose, a list of tasks was established.

The reasoning for the usage of convolutional neural networks is twofold. On one hand, the architecture is extremely popular and its reputation keeps expanding, so aiding researchers on an already established, widely used algorithm seems appropriate. On the other hand, recent work has proven that extracting features from image data is not only possible, but can be achieved in many different ways [46]. Additionally, there is known synergy between this network type and dropout. All these factors guided us towards selecting CNNs as our primary model.

Metalearning's goal is to assist a user to identify the most suitable algorithm(s), given a certain context. It searches for correlations between dataset attributes and the performance of networks [38], enabling a general idea of which approach(es) perform best. It is metalearning's ability to explain that is promising when thinking of regularization technique selection.

In the context of this dissertation, we will be using dropout as our main regularization technique. This choice boils down to its widespread use, popularity, ease of implementation and straight-forward interpretation.

Additionally, this dissertation attempts to verify and build upon the way extracted dataset features are structured in Sonsbeek, 2019. In that work, we believe the statistical meta-features were calculated in a way that induces variance and that may cause significant accuracy drops. To attempt to answer that, we came up with four distinct sampling approaches that should reduce variance and make features more reliable.

In conclusion, this dissertation aims to provide rough guidelines as to how to regularize networks with dropout by obtaining a meta-model that represents knowledge about its effect. It also tries to optimize the process of meta-feature calculation by implementing sampling approaches that boost feature reliability. This helps better predict how relevant said features are to the problem at hand.

1.3 Document overview

The remainder of this dissertation contains the following chapters:

- **Chapter 2, Background** introduces to the reader brief explanations for concepts utilized throughout this work, in an effort to provide the required knowledge to understand the problem at hand. It covers the topics of deep learning, ConvNets, network regularization, meta-learning and its meta-features, gradient boosting and evaluation metrics.
- **Chapter 3, Literature Review** presents related work's progress already made in the area, namely in the fields of meta-learning applied to image data, regularizing with dropout and prior efforts towards neural network explainability.
- **Chapter 4, Approach** proposes a solution to the challenges established before, in terms of applying meta-learning to model dropout. Firstly, we present a general approach for the issue and then focus on how the approach is tweaked in order to fit image data. Research questions for both sections are presented.
- **Chapter 5, Experimental Setup** describes the concrete implementation of the defined approach, explaining the data preparation process, the selected convolutional network and meta-model's architecture.
- **Chapter 6, Results** answers previously defined research questions by critical evaluation of the experiment results.
- **Chapter 7, Conclusions** provides a discussion point for design choices and limitations that overall affect the dissertation, as well as guiding any future work on this work's topics.

Chapter 2

Background

In this chapter, we introduce this dissertation's most relevant topics so the reader may better understand the described work. This includes an introduction to the field of deep learning and its overarching uses, the inner-workings of convolutional neural networks, regularization with special reference to dropout, meta-learning, and more.

2.1 Deep learning

Computers have come a long way to outperform humans in numeric and symbol computation, but have always struggled with more abstract, complex and inherently human tasks like pattern and facial recognition. So, as an attempt to mimic biological neural networks, artificial neural networks were created.

ANNs can be thought of as weighted directed graphs where neurons are the nodes and the edges are the connections between them. There are two main architectural types of ANNs, according to the information's flow: in feed-forward networks, information flows from left to right without loops; whereas in recurrent networks, feedback connections exist.

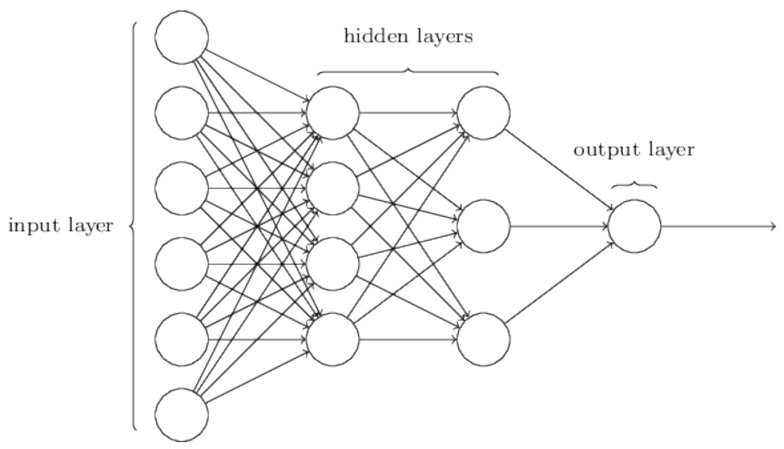


Figure 2.1: An example Multilayer Perceptron model [4]

The multilayer perceptron is the prevalent architecture for feed-forward networks, where neurons are grouped into 3 or more interconnected layers: an input layer, one or more hidden layers and an output layer. MLPs are memory-less, in that computing for a new set of inputs is independent from previous network states. Conversely, recurrent networks are dynamic [26]. Figure 2.1 illustrates an example MLP network diagram.

A neural network learns by updating connection weights according to rules and patterns inferred during training from provided examples – the training set. After a model is designed and hyperparameters are fixed (either manually or by exploring an hyperparameter tuning approach), the network automatically learns patterns on the data. Their ability to automatically establish a ruleset is why artificial neural networks are so appealing for software developers. Because there is no need to hardcode any guidelines or procedures the system should follow, it learns them on its own.

Infering complex concepts out of simpler ones is the basis of deep learning and the reason why even when variance alters data in unexpected ways, networks can still make accurate predictions. For instance, correctly classifying an image of an object even when its shape varies depending on the viewing angle. [24]

2.1.1 Convolutional neural networks

The go-to architecture for image recognition and classification are convolutional neural networks. CNNs have fewer connections and parameters when compared to standard feedforward neural networks, making them easier to train with very little impact on performance [30]. Their ability to make strong and correct assumptions about the nature of images has been proven multiple times across several fields, e.g. detecting and labeling objects and people in images, transcribing images, tracking roads to guide autonomous vehicles, etc.

Convolutional neural networks operate in four different stages: convolution, non-linearity (ReLU), pooling or subsampling and classification (through fully connected layers). Figure 2.2 illustrates an example ConvNet model.

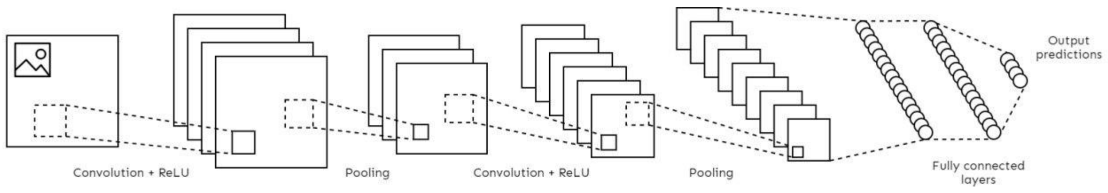


Figure 2.2: An example ConvNet model

As an input, ConvNets use images which have been converted to matrixes of pixel values. Then, a series of convolutions are performed which have the model learning patterns in the data by sliding a smaller matrix – named "filter" – across the input image, computing a feature map. CNNs automatically tweak the filters' values during training although the network's architecture and parameters like number of filters and size have to be explicitly declared. Changing the latter directly influences how many features the network derives and, generally, may result on more accurate predictions. [7] The size of a feature map is managed by three parameters: depth – the number of features used for convolution –, stride – number of pixels by which the filter matrix slides over the image –, and zero-padding – the act of padding the input matrix with zeros around the border.

ReLU (Rectified Linear Unit) is a type of activation function that is linear for all positive values and zero for all negative values. After applying ReLU to an input feature map, the output is a rectified feature map where all negative pixel values have been replaced by zero.

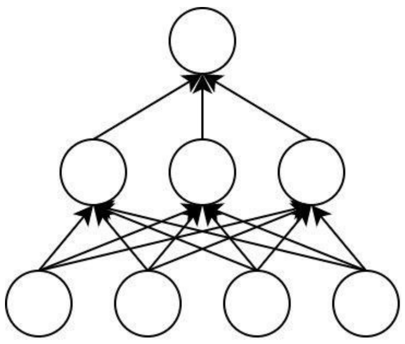
The pooling step downsamples rectified feature maps by compressing the most important features onto a smaller matrix. The two most common functions used in this operation are average pooling – calculating the average for each patch on the feature map – and max pooling – calculating the maximum. Pooling not only reduces the size of the input to a manageable degree, but also controls overfitting [2].

After chaining convolutions and pooling, fully connected layers are attached to the end of the network. Every neuron on a layer is connected to every neuron on the next layer, as traditional on multilayer perceptrons. The high-level features extracted by the previous layers are the now basis for prediction by this smaller MLP. Softmax is the standardized activation function for this step, flattening the classification probability vector to an [0,1] interval and selecting the maximum.

2.1.2 Regularization

In order to better explain the concept of regularization, one must firstly address the fundamentals of overfitting in deep neural networks, as both ideas are closely tied. Overfitting occurs as an indicator of a DNN's lack of generalization capabilities, i.e. a model is overfit when it performs well on training data but fails to make accurate predictions on new data. Thankfully there are well established techniques which aim to constrict these behaviours.

These regularization techniques aim to limit the test error, often sacrificing training accuracy, and come in many forms. The success of each strategy is commonly related to the nature of the machine learning problem and its dataset. Therefore, in practical deep learning scenarios, after


(a) Standard neural network

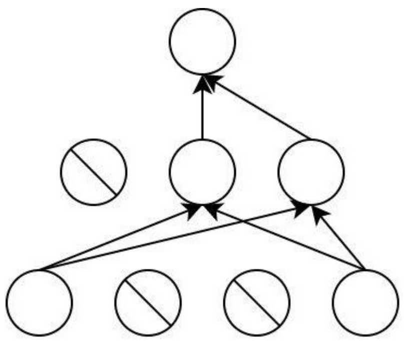

(b) Standard neural network with dropout

Figure 2.3: Neural network diagram with and without dropout applied

appropriate network tuning, it is no surprise the best performant models usually have been suitably regularized. [24]

Perhaps one of the most widely known, powerful and least computationally expensive regularization techniques is dropout [47]. At its simplest form, dropout has the training phase randomly deactivating hidden nodes with a set probability. This mechanism is aimed at removing connections between neurons that depend on eachother. Neuron co-dependency curbs the individual power of each unit and leads to overfitting. [9] As altering nodes and weights during training effectively generates multiple sub-networks, dropout is frequently thought of as training in parallel various neural networks with distinct architectures. Naturally, the base algorithm can be tweaked so it may fit specific scenarios, e.g. using different probabilities for each layer.

Another approach could be decaying parameters that do not contribute significantly towards reducing the objective function to make room for relevant ones to grow. This is the intent of L^2 Regularization, as it is with weight decay. Intuitively, their purpose is to scale weights down in proportion to their current size. The idea behind this strategy is based on large weights introducing instability on the model while, conversely, smaller weights produce subtle, more desirable shifts on it. [45]

2.2 Meta-learning

Metalearning in itself can be defined as *learning the learning process*, hence the self-referencing portion of its name. A metalearning system assumes that extra knowledge may be extracted by reckoning previous experience [12]. Depending on the problem at hand, data scientists may struggle to determine a network's most optimal hyperparameters, often engaging on a trial-and-error methodology, as these choices are analogous to the nature of the data itself. This would mean a certain set of parameters which happen to perform well on a dataset would have to be manually recalculated for a new one. Metalearning aims to overcome this by adapting the algorithms to the problem at hand, by searching for patterns across tasks.

2.2.1 Meta-features

The first step towards effective metalearning is the extraction of dataset characteristics – dubbed "metafeatures" – which hope to correlate with a model's efficacy. It is metalearning's belief that a dataset's morphological characteristics hint towards the task at hand.

In theory, all metafeatures extracted should follow two basic conditions: they should be helpful metrics for algorithm performance evaluation; and they should not be too slow to compute [14]. There are several viable ways to perform dataset characterization:

- **General metafeatures** consist of general dataset information that measure the complexity of the problem.
  E.g. number of observations, attributes, output values, dataset dimensionality.

- **Statistical metafeatures** are derived by performing statistical calculations on the dataset. Mainly appropriated for continuous attributes.
  E.g. standard deviation, coefficient of variation, covariance, linear correlation coefficient, skewness, kurtosis.

- **Information-theoretic metafeatures** are most suitable to characterize discrete attributes, where entropy expresses the dependency of a dataset's attributes and the label. [44]
  E.g. normalized class/attribute entropy, equivalent number of attributes, mutual information of class and attribute, noise-signal ratio, proportion of missing values.

- **Decision tree model-based metafeatures** refer to inducing a decision tree model from a dataset and extracting its characteristics. [3]
  E.g. maximal tree depth, number of leaves, number of nodes, leaf correlation.

- **PCA metafeatures** are gathered by performing principle component analysis and computing principal component statistics.
  E.g. PCA skewness, first PC, PCA kurtosis.

- **Landmarking metafeatures** are computed by extracting the performance of applying simple learners to the dataset. [11]
  E.g. one nearest learner, decision node, naïve bayes.

When deriving metafeatures from image datasets, there are a few tactics which have been proven useful. Naturally, statistical features can and have been used with great success [48]. Yet there are a myriad of possibilities open for exploration. For example, features have been derived from a pre-trained convolutional neural network by extracting the neuron's activity at the penultimate layer [35]. These datapoints are then subsequently applied to other datasets. When tasks for distinct datasets greatly diverge (e.g. mixing banking and health data), adding task-specific metafeatures to the process could be contemplated. [46]

2.2.2 Gradient boosting

Due to the lack of datapoints for the meta-learner (an issue that is addressed later in this document), gradient boosting revealed itself as a viable option.

Boosting is a method of transforming weak learners into strong learners. Michael Kearns describes it as *an efficient algorithm for converting relatively poor hypotheses into very good hypotheses*. It trains a decision tree where each observation is assigned a weight proportional to the difficulty for classification. Larger weights for difficult observations and smaller weights for easy ones. A second tree grows on this weighted data, and the process repeats for a fixed number of iterations. The goal is to improve upon the predictions of previous trees, fixating on the toughest examples with highest errors. Specifically, the Gradient Boosting algorithm uses gradients identify these shortcomings whereas in AdaBoost they are identified by high-weight datapoints. This concept of creating a final model based on the performance of individual models makes gradient boosting an ensemble learner.

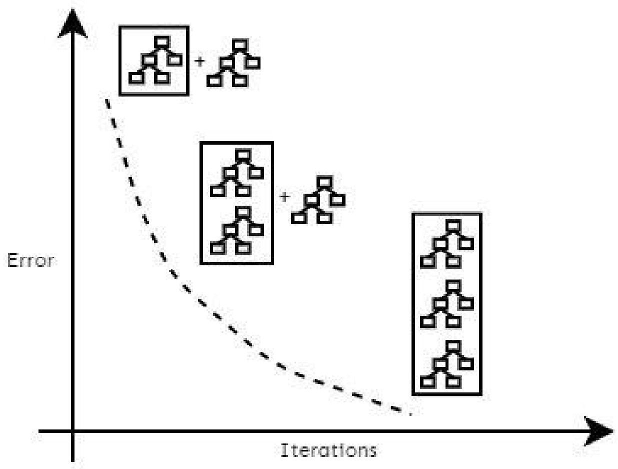


Figure 2.4: Gradient boosting process

XGBoost is an implementation of gradient boosted decision trees focused on performance and flexibility. Its creator, Tianqi Chen, describes it as a self-contained derivation of general gradient boosting algorithm. It allows for model inspection, feature importance analysis and automatic sparse data optimization. XGBoost is also widely regarded as being the fastest open source approach to gradient boosted trees (competing against R packages, scikit-learn, H20, Spark and more) [37]. Because of these benchmarks, it is the go-to framework for winners of data science competitions like Kaggle [8].

2.3 Evaluation metrics

Every machine learning model must be subject to multiple evaluation metrics as benchmarks can widely vary based on the selected metric.

2.3.1 Accuracy, precision and recall

Perhaps the simplest classification measures are accuracy, precision and recall. They all have their use cases but each carries its own caveats.

$$Accuracy = (TP + TN)/(TP + FP + FN + TN) \tag{2.1}$$

Accuracy is the proportion of true predictions among all cases. It is suitable for well balanced and unskewed problems. When it is inbalanced, it provides highly accurate but purposeless results.

$$Precision = (TP)/(TP + FP) \tag{2.2}$$

Precision is the proportion of true positives among all positive cases. Useful when prediction has to be highly rigorous, i.e. it's preferable to miss a few landslide cases instead of triggering false-positives.

$$Recall = (TP)/(TP + FN) \tag{2.3}$$

Recall is the proportion of correctly classified actual positives. Useful when capturing as many positives as possible is a priority. False-positives are welcome here.

2.3.2 F1 score

Accuracy, precision and recall can all be exploited (e.g. recall is 1 if we predict 1 for all examples). F_1 score (also known as F-measure) is the harmonic mean of precision – the proportion of predicted positives which are true – and recall – the proportion of positives which were correctly predicted. It seeks balance between precision and recall while taking into account uneven class distributions. F_1 can also be extended to support different weights and multiclass problems. [1]

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{2.4}$$

2.3.3 Binary crossentropy

Also known as log loss, binary crossentropy is a metric effective for binary classifiers. The uncertainty of a forecast is weighted against the actual label. Like accuracy, it is also sensitive to class inbalances. It's defined by equation 2.5, where p is the probability of predicting 1.

$$-(y\log(p) + (1-y)\log(1-p)) \tag{2.5}$$

2.3.4 ROC curve

The ROC (Receiver Operating Characteristic) curve plots the true positive rate (sensitivity, recall) against the false positive rate $(1 - sensitivity)$. The area under the ROC curve (AUC) separates the positive from the negative classes. When a model has an AUC close to 1, its separability measure is nearly perfect and can safely distinguish between classes. When AUC is 0.5, the network is as good as picking randomly.

Chapter 3

Literature Review

This chapter presents preliminary literature review on two main fronts, which are considered the main focus of this work. Research was made resorting to the keywords *metalearning*, *regularization*, *dropout*, *l1-regularization*, *image segmentation* and *metafeatures*. All papers were found by inserting the keywords into Google Scholar and Semantic Scholar.

3.1 Regularizing with dropout

The literature provides benchmarks available across many regularization techniques. In our dissertation work, we are mostly concerned about the inner-workings of dropout.

Slatton, 2014 has attempted to compare weight decay to dropout in terms of efficiency and most optimal context by training on a smaller subset of the MNIST database. Its main findings were that in situations where dropout performs well, adding weight decay can improve accuracy slightly; and that dropout appears to be most helpful on denser networks, with significant negative results on smaller ones. Conversely, weight decay appears to aid models which are less propense to overfit. These conclusions do not explicitely explain why this is so and the literature urges future work on this area to consider variations to the dropout algorithm and more complex weight decay models.

Srivastava, 2013 set to understand the effects of applying dropout to several datasets with different data sources: image data (MNIST, TIMIT), microphone speech (SVHM), text (Reuters-RCV1) and more. In spite of its notable results, little is known about dropout's averaging, regularization and convergence properties. For instance, the usage of probability $q = 1 - p = 0.5$ for feature detection deletion remains unexplained as to why this specific value is the norm [39]. However, its success in combination with weight decay [45] and max-norm constraints – enforcing an absolute upper bound at each hidden unit by a fixed constant c (with values typically ranging from 3 to 4) – is well documented. As expected, dropout leads to improvements across all datasets. It is hypothesized that this efficiency comes from preventing intra-layer dependency, as each unit is made unreliable. This idea is further supported by inspecting hidden layers' understanding of the data. In networks without dropout, patterns individually do not seem to hold intelligible information whereas, conversely, networks with dropout detect edges and spots. Sparsity is also affected when dropout is applied, as a substantially lower number of neurons fire at the presence of a stimuli when compared with vanilla networks.

Wu and Gu, 2015 set to demonstrate the effects of applying dropout to different layers of a ConvNet – fully-connected, convolutional, max-pooling layers and permutations of these. On the MNIST dataset, a combination of max-pooling and fully-connected layer dropout yielded the least test error, while a non-regularized network performed the worst. Their findings are shown in Table 3.1.

Method	Error %
No dropout	0.81
Fully-connected dropout	0.56
Convolutional dropout	0.60
Max-pooling dropout	0.47
Convolutional and fully-connected dropout	0.50
Convolutional and max-pooling dropout	0.61
Max-pooling and fully-connected dropout	**0.39**

Table 3.1: MNIST test errors for 1x28x28-20C5-2P2-40C5-2P2-1000N-10N trained with dropout in various types of layers [52]

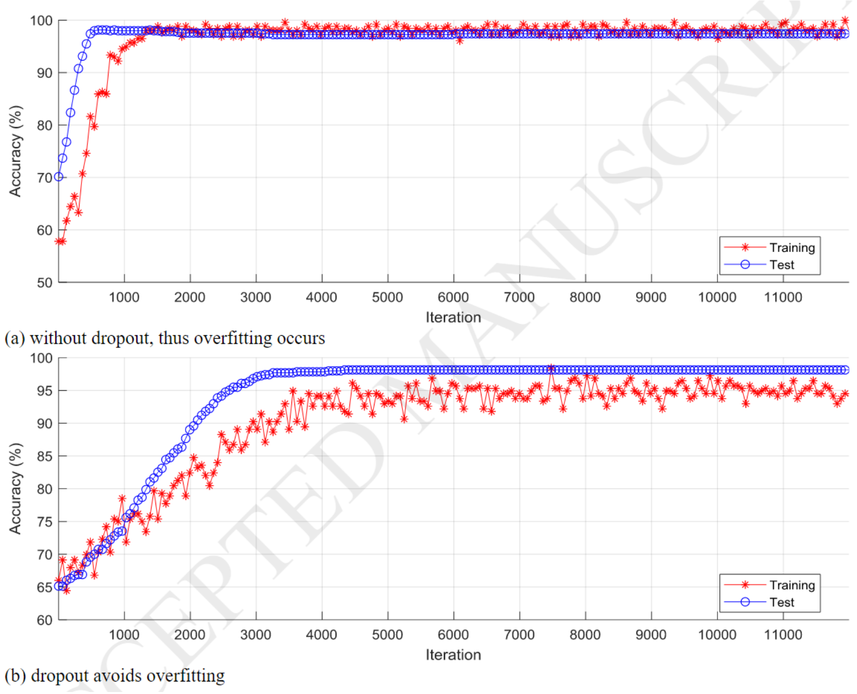
For the CIFAR-10 and CIFAR-100 datasets, results are somewhat identical, showing a clear advantage to using max-pooling dropout allied with FC dropout. Max-pooling only captures the strongest activation in the pooling region, disregarding all other activations, while average-pooling downplays high activations because of it assigns equal contributions to all units. Dropout in max-pooling layers is set to avoid these disadvantages because it introduces stochasticity. Fully-connected layers are always a promising target for regularization because of their big number of units.

The dropout method has inspired multiple authors to tweak and create new models that can outperform dropout. For instance, DropConnect (Wan et al., 2013) regularizes large feed-forward nets by, instead of randomly setting activations to zero, setting a random subset of weights to zero. This makes the fully-connected layers sparsely connected in which its connections are decided in training time. Each hidden unit receives a subset of connections from the previous layer. Its experiments ran on popular image recognition datasets (MNIST, CIFAR-10, SVHN, NORB), achieving state-of-the-art results in most tests. Another example is the stochastic pooling explored by Zeiler and Fergus, 2013. Zeiler and Fergus saw the limitations of dropout on convolutional layers and attempted to adapt the dropout mechanism to them by making the their pooling a stochastic process, instead of deterministic, like the usual average and max-pooling operations. It defends that instead of throwing information away, networks should select from what it is already providing.

Based on a multinomial distribution of activations within a pooling region, a single pooled map response is selected. The model shows state-of-the-art performance when applied to multiple common datasets: MNIST, CIFAR-10, CIFAR-100 and SVHN.

Ba and Frey, 2013 takes inspiration from the concept of dropout by creating a binary belief network that stochastically adapts its architecture based on input. The result is a network that, instead of randomly switching neurons off and on, does so depending on input activities. This approach comes from the authors' concern that dropout turns off highly confident hidden units half of the time. Overall, this standout network appears to noticeably outperform standard dropout networks on the MNIST and NORB datasets, and even deeper and more complex models. These results seem very relevant in the context of this dissertation, given the fact that both projects limit themselves to convolutional neural networks.

Zhang et al., 2018 aims to detect multiple sclerosis by developing a ConvNet model that could train on a balanced dataset of images of healthy and unhealthy brain slices. Applying dropout to the network yielded an accuracy boost of 0.88% compared to a network without any regularization. Their approach follows the common practices for regularizing convolutional networks with dropout: it is used before fully-connected layers (rather than on convolution layers) and with an average retention probability of 0.5. In fact, their model includes 7 convolution and 3 FC layers and dropout is applied to each of the latter (with retention probabilities of 0.4, 0.5 and 0.5, respectively). Remember that retention rate is the reverse of dropout rate, so in fact the first fully-connected layer is dropping over half of its hidden units during training.



(a) without dropout, thus overfitting occurs

(b) dropout avoids overfitting

Figure 3.1: Model training and test accuracies for networks with and without dropout

Figure 3.1 illustrates the prevention of fully-connected layers' weight co-adaptation by employing dropout. In conjunction with parametric ReLU, an activation function that improves model fitting with low overfitting risk, Zhang accomplished state-of-the-art results in comparison to other multiple sclerosis detection systems.

Study	**Dropout type**	**Dropout rate(s)**
Slatton, 2014	All layers	$p = 0.5$
Srivastava, 2013	Only in FC + all layers	$p = 0.8$ (1^{st} FC), $p = 0.5$ (rest)
Wu and Gu, 2015	Mixed	$p = 0.8$ (1^{st} FC), $p = 0.5$ (rest)
Wan et al., 2013	Custom	n/a
Zeiler and Fergus, 2013	Custom	n/a
Ba and Frey, 2013	Custom	n/a
Zhang et al., 2018	Only in FC	$p = [0.6, 0.5, 0.5]$

Table 3.2: Study comparison regarding dropout's experimental setup

In conclusion, through analysis of the study comparison Table 3.2 regarding the studies' dropout experimental setup, we may recognize a few patterns and establish some ground rules:

- Dropout is capable of being applied to both/either convolutional and/or fully-connected layers and both are able to produce positive results. However, there are still a few caveats for applying dropout to convolutional layers and is still sought to be unsuitable by a few authors [21]. Based on this reasoning, we resorted to strictly applying it to FC layers;
- Regardless of selected layer, dropout rates tend to hover around value $p = 0.5$. This prompted us to also use this value in this dissertation;
- Dropout does not need to be strictly followed, as multiple researchers have borrowed from dropout's base idea and implemented their own regularization approaches with success.

3.2 Extracting metafeatures from images

The process of extracting metafeatures from images is challenging due to the lack of literature on the matter. Most techniques are aimed at tabular data and not image data specifically.

Sonsbeek, 2019 advances medical image segmentation state-of-the-art through a metalearning approach. It proposes a system that automatically selects a deep learning model fit to solve a specific problem, illustrating how past performance of algorithms combined with meta-information of a new dataset yield valuable predictions. It trained and validated on medical datasets pulled from the Medical Segmentation Decathlon (MSD) challenge. All datasets include imagery of different human body regions captured through a pair of modalities (either magnetic ressonance scans or computed tomography) and vary from dozens to a few hundred instances on each train and test sets. When deriving features from the datasets, the authors underlined three different approaches. Firstly, statistical metafeatures describe the numerical properties of a distribution of data. A few examples are a dataset's number of instances or other measures used to describe distribution, like skewness, correlation, sparsity or kurtosis. Secondly, convolutional neural networks operate by performing automatic feature extraction on data to make predictions. So deep learning metafeatures are automatically fetched by any CNN architecture albeit at the cost of holding unclear meaning. One dataset's features are then correlated with others' in an attempt to emulate how, in statistical datasets, two datasets can be compared through their data distribution. Finally, sometimes datasets' particularities hinder feature extraction due to how widely different the nature of the data is. Task-specific metafeatures identify these divergence aspects. In the medical image segmentation scenario, that would mean introducing metrics like whether data is captured through CT or MR scans.

Naseer and Zafar, 2018 successfully used both convolutional and long short-term memory networks to train on a dataset consisting of a cursive script language called Urdu in hopes of recognizing characters within ligatures of distinct font sizes. Metafeatures are extracted from ligature thickness graphs, models inferred from the dataset, effectively categorising the metafeatures as model-based. It achieved a network performance ranging between 90% and 99.8% and the average performance when using meta-features surpassed the usage of raw images by 1.01%.

Lorentzon, 2017 investigates features used for selecting images which are worthy of further analysis, according to three different measures: having good quality, salient content and being unique. The work makes use of the Common Objects in Context (COCO) dataset, which contains over 200k labeled images across 91 categories such as food, vehicles, furniture, domestic appliances and cutlery. The paper makes the distinction between iconic images – the subject is centered and in canonical perspective – and non-iconic images – contain contextual information and subject is in non-canonical perspective. Three methods for feature extraction are considered. Histogram of oriented gradients (HOG) is a feature descriptor used for the purpose of object detection where an image is divided into small connected regions and, for all pixels in each, a histogram of gradient directions is computed. Representing an image in the discrete cosine transformation domain (DCT) is concentrating most of the visually meaningful information in a set of coeficients. A convolutional neural networks' (CNN) convolutional layers automatically perform feature extraction via filters which detect patterns in the data and interpret it in a unique way. Their output is then sent to the fully connected layers where prediction happens. This work's selected predictor is a support vector machine (SVM) which functions by laying down an hyper-plane that isolates one class from the other, maximizing the margin between the center-most points on both classes. In the end, DCT alongside the SVM classifier was helpful is distinguishing between good and bad quality images, while CNN performed well when separating salient from non-salient images.

Campos et al., 2016 extracts a set of 44 features from images of four different meta-databases to create a meta-recommending system for image segmentation algorithms. This set of features is based on histograms, contrast and quality, gray-level co-occurrence matrixes, Fast Fourier Transforms and the statistical information of color from RGB (red, green and blue) channels as well as HSV (hue, saturation and value) channels. Through analysis of attribute importance, although color features from both color spaces appear very meaningful, no feature outperformed through all meta-databases, suggesting that meta-feature selection may not be fitting for this image segmentation problem.

Davis et al., 2012 applied feature extraction to hand radiograph images to generate a predictive model of bone age. Size descriptors like height and width of hand bone portions are naturally good indicatives of age, so after computing region-of-interest boxes, the dimension and shapes for bones like the phalanx and the epiphysis are extracted. Although extracting task-specific representations like these is optimal for a focused problem like predicting age from radiograph images, the general nature of our meta-database constrains us to use more broad features.

Study	**Meta-feature type**
Sonsbeek, 2019	Statistical, information-theoretic, model-based, task-specific
Naseer and Zafar, 2018	Model-based
Lorentzon, 2018	Model-based
Campos et al., 2018	Statistical, information-theoretic
Davis et al., 2018	Task-specific

Table 3.3: Study comparison regarding the type of extracted meta-features

Overall, existing literature proves that extracting meta-features from images is not only possible but can be accomplished in many ways – as seen in Table 3.3 – with solid results. However, it has to be noted that deriving task-specific features – features conceptualized manually that are unique to one or a set of datasets, e.g. information as to how an image was captured – are infeasible because of the goal of generalizing the model as much as possible. The rules set by the meta-model may not be applicable to new datasets. For instance, a feature that identifies interest zones in medical imagery is not compatible with datasets from other fields.

3.3 Deep networks explainability

Models can be *ante-hoc* when they are designed to be inherently explainable (e.g. logistic regression, decision trees), or *post-hoc* where the explainability objective is added after training. Schaaf and Huber, 2019 focuses on optimizing deep MLPs towards post-hoc decision tree extraction. The work makes use of $L_1 - O$ regularization to improve the extraction of decision trees from deep neural networks and reduce model complexity. Pruning the decision trees just enough appears to provide information in a more meaningful and concise way instead of condensing all information to a single node. It alerts to the need of establishing a trade-off between model complexity and comprehensibility.

Lee et al., 2018 set to create a deep-learning algorithm capable of detecting acute intracranial haemorrhage from small datasets. The authors were aware of the legal requirements for clinical decision support software to explain the reasoning for their decisions and that the medical field remains uneasy with deep learning's opaque design, one of its biggest hurdles preventing it from widespread adoption. With this in mind, aggregated with their model, a visualization tool was developed which displays the basis of the predictions. It sorts the feature maps from all convolutional layers by their maximum activation values and overlays a heatmap that highlights the most important regions for the algorithm.

Finally, Liu et al., 2018 aims to achieve interpretable CNNs through a meta-learning approach by taking a single hidden layer – the first fully-connected layer – and learning about the patterns it holds. Similarly to Schaaf and Huber, 2019, the methodology follows a *post-hoc* approach. Its meta-learning approach is somewhat similar to the route taken in this dissertation: although its final purpose is different, we also feed the meta-level training data based on a type of landmarking into a tree-based algorithm (random forest in their case, gradient boosting in ours). It displays a visual result to indicate whether a test instance was correctly classified by checking whether there are any overlaps in corresponding activations. The similarity of this work's architectural choices motivates us to be confident in our meta-learning with landmarking approach.

Chapter 4

Approach

This chapter introduces the concept of modeling dropout with meta-learning with a general and an image-specific approach, regarding meta-feature selection, the process of performance estimation and creating a meta-model.

4.1 Modeling dropout with meta-learning

The general method for modeling dropout with meta-learning involves (1) generating meta-data by characterizing datasets through a certain representation (meta-features); (2) executing performance estimation on the datasets across networks with variable dropout rates; and (3) using both the extracted meta-data and performance to create a recommendation system for which dropout rate should work best on a provided new dataset (a meta-model). An example diagram for this system architecture was made available in Figure 4.1.

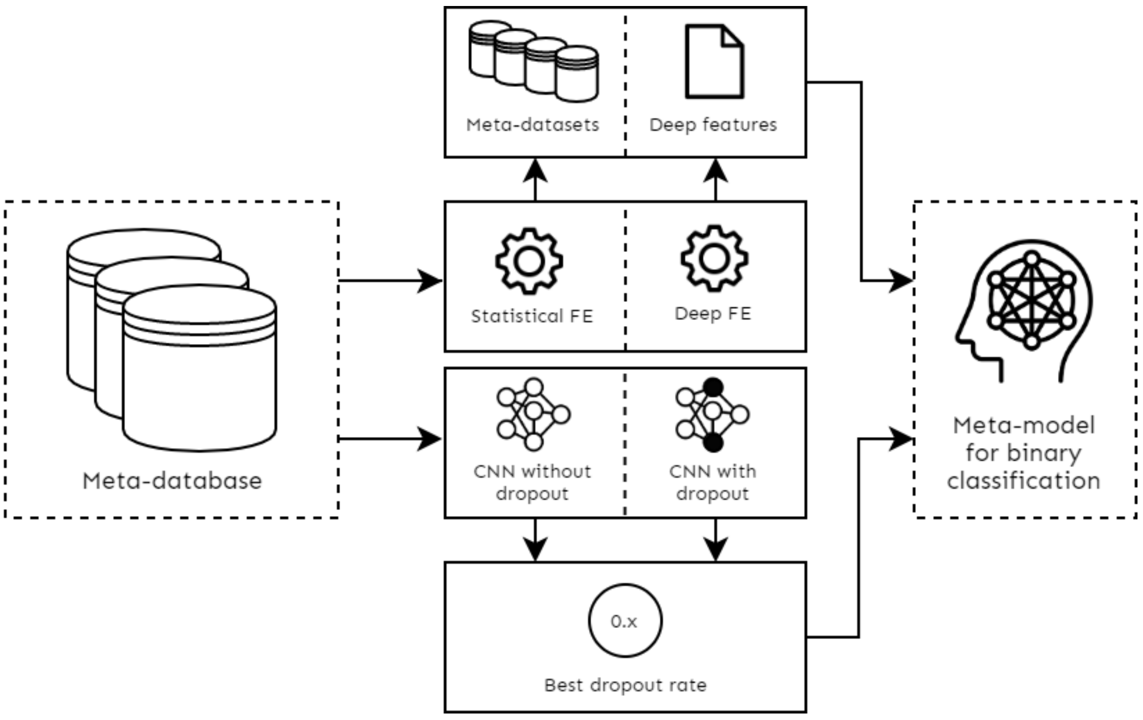


Figure 4.1: System architecture for modeling dropout with meta-learning

4.1.1 Statistical meta-feature selection

In meta-learning, datasets must have their characteristics extracted in some way. This follows the foundation on which machine learning algorithms operate: isolate the most meaningful patterns and make predictions according to them.

Characterizing a dataset can be done in numerous ways, as described in section 2.2.1. In this dissertation, we have decided on two approaches: statistical and deep meta-feature extraction.

Table 4.1 presents the set of 29 statistical measures used to describe a dataset. The list is a result of literature review of commonly used meta-features in meta-learning [13, 38, 46]. Each feature underwent explorative analysis to prevent redundant elements.

4.1.2 Deep learning meta-feature selection

An upside of performing deep feature extraction is doing so automatically where, in contrast, statistical features must be individually thought of. While the latter generally holds a clear meaning because they are the result of calculating simple and well-known features, deep learning features naturally dwell in the black-box space. Nevertheless, even if these features are of puzzling interpretation for any human, as proven by ConvNets' effectiveness, they can be useful, provided that learned representations can be transferred between datasets. Even if statistical analysis of a dataset is very surface level, a deeper analysis at the cost of explanability is sometimes preferred.

In this dissertation, we have made use of well known ConvNet classification models. All these canned architectures were imported from Keras and are loaded with weights pre-trained on the

Statistical meta-features	
1	Mean pixel value
2	Standard deviation of pixel value
3	Coefficient of variation of mean pixel value
4	Mean skew value
5	Standard deviation of skew
6	Coefficient of variation of skew
7	Mean kurtosis value
8	Standard deviation of kurtosis
9	Coefficient of variation of kurtosis
10	Mean entropy value
11	Standard deviation of entropy
12	Coefficient of variation of entropy
13	Mean median value
14	Standard deviation of median
15	Mean mutual information value
16	Standard deviation of mutual information
17	Coefficient of variation of mutual information
18	Max mutual information
19	Mean correlation value
20	Standard deviation of correlation
21	Coefficient of variation of correlation
22	Mean sparsity value
23	Standard deviation of sparsity
24	Coefficient of variation of sparsity
25	Mean XY axis value
26	Standard deviation of XY axis
27	Coefficient of variation of XY axis
28	Equivalent number of features
29	Noise signal ratio

Table 4.1: List of statistical meta-features used

ImageNet database. Networks pre-trained on ImageNet are capable of classifying images into 1000 object categories. Below follows a list of models used in this dissertation, alongside a quick overview:

- VGG16 consists of 16 weight layers, including 13 convolutional and 3 fully-connected layers, uses (2x2 and 3x3) filters, totalling to approximately 138 million parameters;

- VGG19 is slightly deeper than its previous iteration, consisting of 19 weight layers, including 16 convolutional layers and 3 fully-connected layers;

- ResNet-50 is 50 layers deep and innovates the field with the introduction of the skip connection – a way to feed primary layers' information directly to the later layers without it turning too abstract for further interpretation;

- MobileNetV1 is 53 layers deep and provides state-of-the-art accuracy while requiring as little memory and computing power as possible. It makes use of inverted residuals and linear bottlenecks.

Similarly to statistical meta-features, we sample multiple times from a single dataset and extract features from each. In some datasets there might not be enough images to completely fill a stipulated sample size. In those cases, the data is padded with zeros until it matches the expected size. Due to the small sample size for statistical features, this issue should not happen for the statistical extractor unless the provided dataset has less than 16 total images for training.

4.1.3 Performance estimation

Performance estimation can be defined as a meta-feature extraction technique that tries to determine position of training data in the areas of problem learning by directly measuring the performance of learning algorithms themselves [11].

In the context of this dissertation, performance estimation is used to feed our meta-model with the target labels of our meta-database. If our overall goal is to predict which retention factor works best on a network that would train on a new dataset, it is evident that we should actually be calculating, for our meta-database, which rates work best and feed that knowledge to the model.

However, doing the step of performance estimation on regularization is subject to some decisions that may very well influence the final model's predictive power. Note that regularization affects data differently and non-linearly, meaning that applying dropout to some dataset can change the result slightly, greatly or by nothing at all. Additionally, these outcomes are variably dependent on the contents of the training split, so there is a need to legitimize the performance data by training and predicting on the model multiple times to infer exactly how dropout affects the dataset in question.

To exemplify this issue, take the *basic-shapes* dataset, which is comprised of 300 28x28 images of three distinct shapes: circles, squares and triangles (100 images for each class). Its balance and tiny total size allows for speedy training times and quick testing.

DR	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	avg
0.0	0.3	0.334	0.567	0.4	0.6	0.5	0.5	0.367	0.467	0.467	0.45
0.25	0.5	0.567	0.433	0.567	0.433	0.567	0.567	0.367	0.567	0.533	**0.51**
0.5	0.5	0.367	0.533	0.433	0.533	0.6	0.5	0.5	0.367	0.533	0.486

Table 4.2: Predictive performance of 3 networks (with dropout layer set to 0.0, 0.25 and 0.5, respectively) across 10 repetitions on the *basic-shapes* dataset

Table 4.2 illustrates the predictive performance of 3 networks whose only architectural difference is how agressive the second to last fully-connected layer is regularized. The first has zero dropout rate and connections pass-through as it there was no intermediate layer at all; the second has a dropout rate of 0.25, meaning that 25% of the neurons are shut off; and the last has the commonly used rate of 0.5, where half of the connections from the first FC layer to the final softmax one are lost. Remember that we are not looking as much for impressive accuracy, as this model is specifically fine-tuned for MNIST and should not behave optimally for our entire set of datasets. Instead, we are paying close attention to patterns that signal that one network is performing better than the others. As we can see, at first glance, no dropout rate appears to be a clear winner. The best we can do is average out all accuracy values and name the maximum the preferred rate. The more times a single network is tested, the more sure we are of our choice. However, even if *basic-shapes*' small size allows for even a hundred runs for each network, the meta-database contains datasets surpassing gigabytes of size on disk where even performing three laps across all networks is painfully slow. So there is most certainly a compromise of the training labels reliability and compute time and finding the balance between this trade-off is key.

It is also important to understand that achieving state-of-the-art results on model accuracy does not fit the scope of this project. We are only concerned with substantial divergences in accuracy and so, within reasonable limits, disregard the actual model score. Even if we actually were seeking state-of-the-art performance across all datasets, it would be unfeasible in this generalized fashion when considering the necessary hyperparameter tuning unique to each dataset. The 'No Free Lunch' theorem confirms this concern: it states that there is no one model that works best for every problem. If a learner A performs better than learner B in select circumstances, then learner B must outperform A on other instances [38]. Additionally, tailoring networks to maximize accuracy might place us in a situation where, after running the experiments, it is not evident whether a dataset benefits from agressive dropout rates because of its characteristics or because its fine-tuned model just usually benefits from higher dropout rates. To eliminate this possibility, we have fixed one standard convolutional neural layout network and applied it to all datasets.

4.1.4 General meta-model

In its most simple state, a meta-model consists of an objective, a learning algorithm and optimizer, and dataset metadata. Let us define what a meta-model should be comprised of when applied to this context:

- The model's **objective** is to make predictions on which dropout rate is the most desirable through statistical classification;
- As long as it allows for converging on the objective, any supervised **learning algorithm** could work. Gradient boosting, decision trees, support vector machines (SVM), neural networks (multilayer perceptrons) and more are valid when justified;
- The meta-model must have **dataset metadata** in the form of features that can identify the data. Its target variables should refer to the best or range of best dropout rates that yield the best performance for a dataset [20].

Building a meta-example is a two-step procedure where two processes are executed on top of a dataset in order to refine it into a feature-label tuple (x, y). Feature x is the dataset's data characterization, its meta-features inferred from either a statistical or deep technique. Label y is a numerical value corresponding to the dropout rate that, on average, across several executions of classification models that both lightly and heavily employ it, got the best accuracy.

This problem may be either be modeled as a classification problem or as a regression problem. When in the loss function $f : x \rightarrow y$, y is a discrete variable (class labels), it is a classification problem. If y takes continuous values, it is a regression problem. A dropout layer's rate value is only valid in the interval $[0.0, 1.0[$ and while it is unusual to apply very specific dropout rates (floats with more than two decimal places), there is no theoretical reasoning as to why it should not be valid. So while this can be modeled as a regression analysis problem, our version of the meta-model is performing binary classification – determining whether a dataset belongs to the 0.0 or 0.5 class.

This motivates us to formulate our first research question:

- **RQ1**: Can a meta-learning approach be used to predict the effect of dropout?

4.2 Approach for image classification

An image classification approach to this meta-learning problem roughly follows the steps above. However, the type of data lifts a couple of complex issues when compared to tabular data.

Firstly, the challenge of meta-feature characterization of image data is evident. Not only the literature is fairly limited on extracting meta-features from images, but the few documented cases have not been used in such a general context as in here. This corners us into extracting representations of images that are applicable to all datasets. The feature extraction step is definitely crucial and motivates our work, however we do not address it in depth, having limited ourselves to analysing features that were proposed by others.

Secondly, it is close to impossible to utilize entire image datasets in our experiments, due to the volume of data. Therefore, the obvious approach would be to randomly sample from datasets and let meta-features be calculated from a smaller (and hopefully representative) subset of images. To put this issue of data size into perspective, when we perform automatic feature extraction with the ResNet50 encoder – whose architecture consists of a total of 2048 filters –, performing 16 samples of 16 images resulted on a features file with size over 250 MB for a single dataset. While statistical meta-features do not suffer from heavy memory usage, computation times can reach up to 1.5 hours for tiny samples that require image correlation calculations. This is a factor that grows exponentially with the number of images added to the mix. Also one must consider how datasets vary in size, not only in terms of disk space, but also examples and classes. Sampling ensures the data characterization for all datasets is balanced, avoiding having very few features for smaller datasets.

4.2.1 Statistical meta-feature extractor

In its most basic form, the statistical meta-feature extractor computes an array of 29 meta-features for a provided set of images. Both the number of samples and number of images in each sample are adjustable parameters. This statistical characterization includes values that range from averaging pixel values to measuring the difference between image shapes within a subset.

As explained, extracting features from the dataset as a whole is computationally unviable. But including more images beyond a certain threshold appears to follow the law of diminishing returns, where the benefits in accuracy gained are less than the computational time and energy invested. We may not need terabytes of data to estimate a parameter or test an hypothesis provided that the selected arbitrary set of rows preserve the features of the original data [33].

However, we believe the sampling process has a big say on the overall predictive power of the algorithm. On Sonsbeek, 2019, the meta-feature output is simply a list of concatenated samples. In this dissertation, we performed further data transformations in hopes of reducing training error and eliminating any positional correlation between features inside samples. The statistical meta-feature extractor is described below, alongside the four generated meta-datasets and the steps required to form them.

If S is the full image training set from a given dataset, then we denote by $[S]^k$ the set of k-subsets of S that is $[S]^k := \{X | X \subseteq S \land |X| = k\}$. In the context of neural networks, this is similar to partitioning data into mini-batches of size k but only keeping $|[S]^k|$ batches. Applying this sampling procedure to all image datasets generates a set of 50 $[S]^k$ sets.

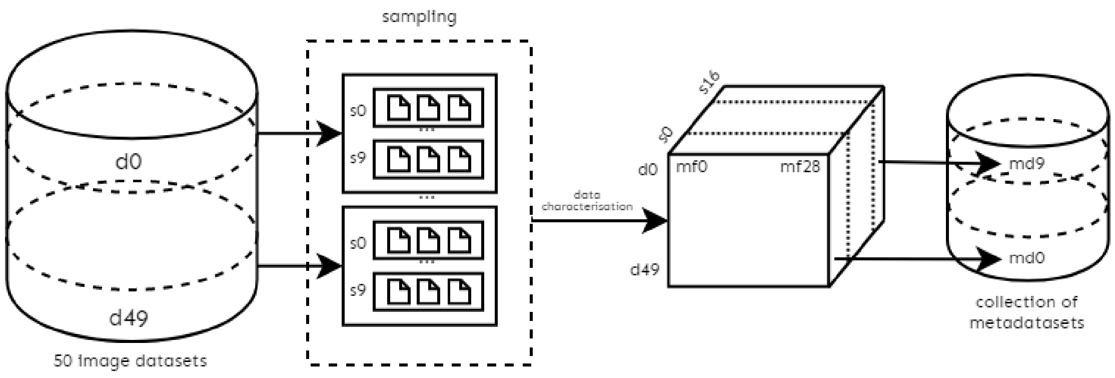


Figure 4.2: System architecture for statistical extractor of meta-dataset #1

Then, from each sample in $[S]^k$, an array of 29 meta-features is derived. In brief, each dataset has samples and each sample has meta-features. An intuitive representation of this data hierarchy is a tensor (also referred to as 3D matrix or tridimensional array). Figure 4.2 features a cube relating these concepts for easier interpretation. Let it be denoted as F.

In a first approach, F is sliced once for each number of samples determined in the sampling process. Each slice G is expressed as $G = \{D_0, ..., D_{49}\}$, where D_i is itself a meta-feature set $D_i = \{MF_0, ..., MF_{28}\}$ and produces a meta-dataset. As such, there are as many meta-datasets as the number of samples extracted from each dataset. This is virtually equal to the approach used by Sonsbeek, 2019 and therefore is our control meta-dataset. The only notable distinction is that here, datasets are concatenated to generate a meta-dataset whereas in their work, each dataset was saved separately.

The second approach – illustrated in Figure 4.3 – makes use of the prior experiment's 3D matrix F. Instead of slicing F across the sample Z-axis, let each cut G now be done on the meta-feature X-axis that relates datasets to samples. Thereafter, a fixed number of shuffle operations are applied to each slice G. Now this transformed cube F' constitutes a meta-dataset and all subsequent shuffles are new meta-datasets. So, there are as many meta-datasets as the number of shuffles performed to each slice. This mechanism of interchanging samples within a dataset's meta-feature is aimed at eliminating any positional correlation.

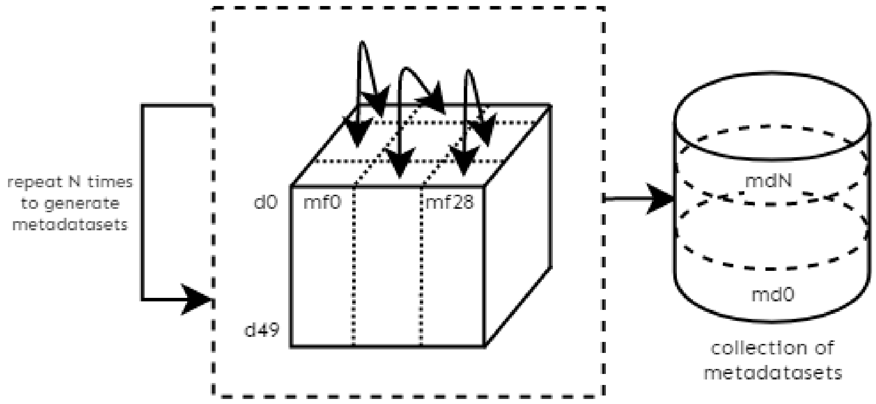


Figure 4.3: System architecture for statistical extractor of meta-dataset #2

Analysing the previous work's approach motivates us to formulate this dissertation's second research question:

- **RQ2**: Does the sampling approach proposed by Sonsbeek, 2019 induce variance in the data?

4.2.2 An approach to reduce variance

Motivated by our expectations of variance obtained in the first two experiments, we now propose a new approach to reduce variance.

For this third approach – illustrated in Figure 4.4, the collection of transformed matrices F' is the baseline. The goal of this procedure is to flatten the sample dimension by averaging out and computing the standard deviation across all samples.

This is accomplished by taking each tensor F' and slicing it in the same fashion as to Figure 4.3 – so that each slice G is a shuffled set of samples for a given dataset and meta-feature. Then, sample from that set and compute its average and standard deviation. Finally, replace the original sample dimension with the results of the calculations. Doing this across all datasets and meta-features effectively flattens the Z-dimension in F'. Similarly to approach 4.3, there are as many meta-datasets as the number of calculations performed. This approach attempts to simplify the feature set by aggregating all samples into two values and avoid redundancy in the data.

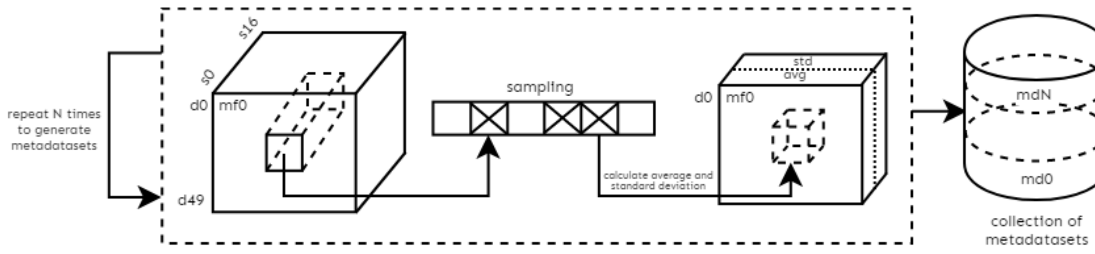


Figure 4.4: System architecture for statistical extractor of meta-dataset #3

A final 4th approach is based on the 3rd one but instead of sampling from the set, compute the mean and standard deviation from its entirety. This results on a flattening of the sample dimension. There is no need to repeat the process like before because the output would just be the same.

Investigating these approaches to trying to reduce sampling-induced variance allow us to formulate this work's last research question:

- **RQ3**: Does the aggregation of features obtained on samples using average and standard deviation reduce the effect of variance?

Chapter 5

Experimental setup

This chapter describes the concrete implementation of the defined approach, explaining the data preparation process, the selected convolutional network and meta-model's architecture.

5.1 Data preparation

This meta-learning approach involves collecting as many image datasets as possible, characterizing them and measuring their performance when applied to convolutional neural networks with varying dropout rates. But additionally, in image classification, plenty of image data – a data type that is heavy by design – is needed as well. In non-meta-learning problems, typically a single dataset with various features and examples is used. However, in meta-learning, an entire dataset constitutes a data point.

5.1.1 Standardizating input

In a single dataset problem, data preparation procedures are selected to fit the nature of the data. Conversely, a meta-learning problem expects the datasets to fit a standardized procedure.

The 50 datasets used in this dissertation were mostly downloaded from Kaggle and TensorFlow Datasets. But no single platform hosts datasets in a consistent way. Datasets can lack labelling, making them only feasible for unsupervised learning tasks. Data can be explicitely partitioned into train, test and validation splits or can require later manual splitting. The folder organization may fluctuate as well, with some datasets arranging the splits in folders while others resort to renaming files to illustrate to which split or class the image belonged to. Image file formats frequently vary, sometimes within datasets, which is troublesome when considering decoding an image in TensorFlow, at the time of writing, is limited to BMP, GIF, JPEG and PNG extensions. Data can be encoded in non-standard formats such as Numpy arrays, CSV spreadsheets or compression archives like TAR packaging or GNU zips. Additionally, usually Kaggle's user-submitted datasets frequently have corrupted files or system files (like macOS' .DS_Store) that can halt any subsequent methods if not manually detected and removed. More often than not, we found that an otherwise valid dataset could not be selected because of its size (either too large or too small) or subject. All these variables made the data preparation period challenging and lengthy.

The approach was to individually tackle and automate these issues if a large enough number of datasets shared the same problem. Therefore, the resulting preprocessing script automatically cuts datasets according to missing splits, converts all TIFF images within a dataset to JPEG and inserts classes into folders conforming to regular expressions manually written for each dataset. It is also possible to download datasets from TensorFlow Datasets from this tool.

5.1.2 Data characteristics

A data science platform like Kaggle makes available a handful of image datasets that fit the context of this dissertation. However, a prerequisite for including a dataset in the meta-collection, apart from being compatible with our setup, is ensuring they all cover the widest range possible of dataset properties. For instance, it is relevant to include datasets with balanced and skewed class distributions, datasets aimed for binary-class and multi-class classification and datasets with few and plentiful examples.

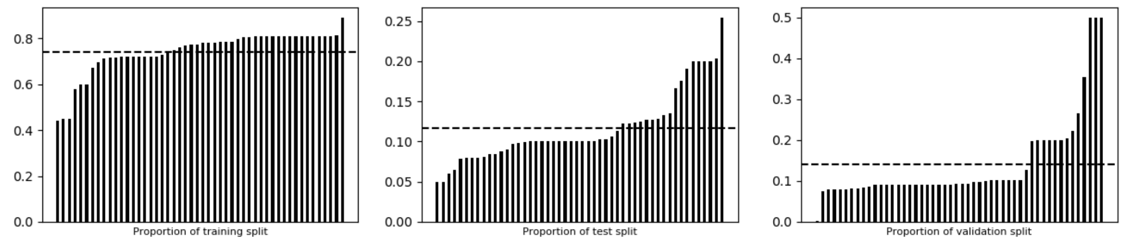


Figure 5.1: Meta-dataset split distribution

The 50 datasets' split distribution is plotted on Figure 5.1. Original splits were maintained when available. Otherwise, the data is shuffled and split according to an 80% training, 10% test and 10% validation distribution. Unavailable splits are always pulled from the training set. By analysing the plot, we consider the range of selected datasets' split distribution well varied.

Class imbalance plays a huge role on the predictive power of a model because most machine learning algorithms assume an equal number of examples for each class. We denote by $C = \{x_0, ..., x_n\}$ the set of a dataset's classes. Then $\forall x \in C.C_x = C_x / \sum_{i=1}^{|C|} x_i$ transforms C into a list of class proportions. Finally, get the standard deviation of C. This constitutes an imbalance ratio that

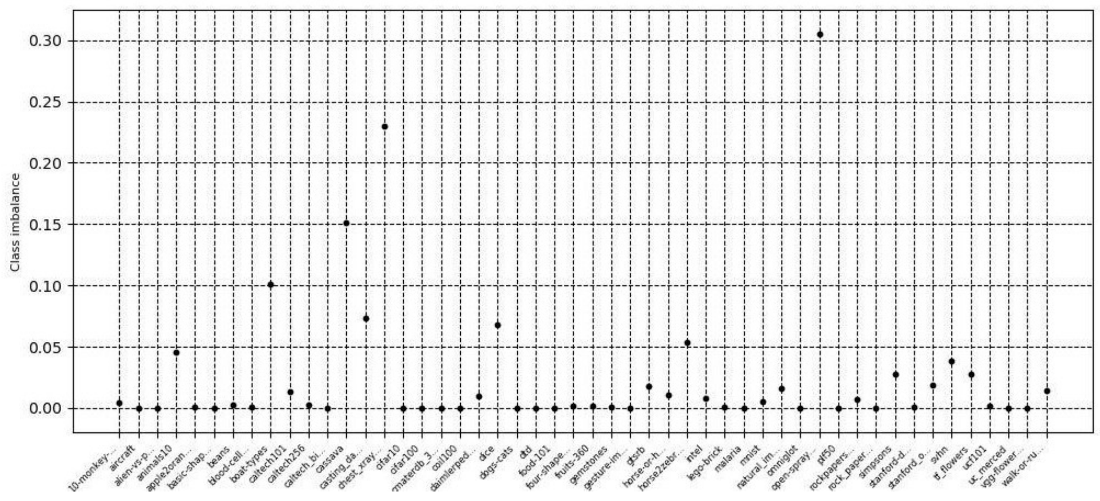


Figure 5.2: Meta-dataset class imbalance

enabled us to evaluate how much our meta-dataset suffers from class imbalance. The imbalance ratio across all datasets has been plotted on Figure 5.2.

Below follows a list of dataset morphological characteristics:

- Regarding the number of classes, 10 out of the 50 datasets (20%) are binary (2-class) classification datasets, with the remaining 40 (80%) being multi-class classification datasets. Out of these 40 datasets, 19 have a class count fit inside interval $[3,10]$, 13 are in interval $]10,100]$ and the remaining 8 have over 100 classes.

- A significant number of meta-features require datasets with images of varying sizes. 48% of the datasets have images with unique dimensions, while the images on the remaining 52% datasets all have identical proportions.

- Overall, only 7 out of the 50 datasets have over 5% class imbalance ratio. The worst offenders are the *open-sprayer* and *chest_xray_pneumonia* datasets, rounding to 30.5% and 22.9% class imbalance ratio respectively. This means that, for instance, on the *open-sprayer* binary-classification dataset, one class has roughly four times the number of examples of the other's. The rest of the datasets are better distributed, with 32 having less than 1% IR and 15 being perfectly balanced.

Appendix A lists all datasets used in this dissertation and their full specificities.

5.2 ConvNet architecture

The chosen convolutional neural network follows the common pattern of stacking 2D convolutional layers and max pooling layers. The convolutional base is the same used by Google's ConvNet tutorial found on the official TensorFlow documentation [6] and appears to be built specifi-

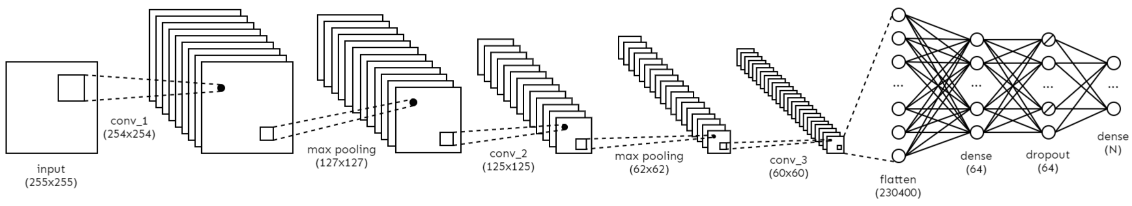


Figure 5.3: Base convolutional neural network architecture

cally for the CIFAR-10 dataset. It is comprised of three convolutional layers – one with 32 filters and two with 64 filters – alternating with two max pooling layers. The network is then flattened and suffixed with a couple of densely-connected layers: the first outputs 64 dimensions while the final layer has as many outputs as the number of dataset classes. Activation functions are ReLU and softmax respectively. Finally, our change to the model's overall architecture is an added dropout layer placed in-between the final dense layers. By applying the same network across all datasets, we hopefully ensure meta-level predictions are unrelated to changes in the network's architecture.

The placement of the dropout layer was chosen after extensive literature review on the matter. However, applying regularization to other layer types has been proven effective [52]. So when presented with the choice, we opted for the most popular way of regularizing.

During the deep learning process, the model is frequently recompiled because of two changing variables: each dataset's class number affects the last layer's prediction dimensionality (illustrated as *N* in figure 5.3) and experimenting with dropout rates affects the dropout layer.

5.3 Meta-feature extractors

For all meta-feature extractors, either statistical or deep, for each dataset, its features are the result of calculating features from a set of 16 samples, each comprised of 16 images. Regarding statistical extraction, in experiments that require multiple repetitions (demonstrated in Figures 4.3 and 4.4), they are performed 10 times. Training sets are always shuffled before any operation.

It is important to verify whether meta-features are meaningful to the model. It is proven that redundant and correlated features slow the training process, potentially harming the performance of the model and making it harder to interpret.

Figure 5.4 illustrates the dispersion of meta-features across datasets. This is particularly useful for understanding how relevant a given meta-feature appears to be for prediction. In theory, a higher coefficient of variance should be correlated to a more useful feature. For instance, in one hand, meta-features related to sample skewness (indexes 3-5) and noise signal ratio (index 28) do not deviate too much, supporting the possibility that image asymmetry is not too useful to the model. On the other hand, mutual information (indexes 14-17), image correlation (indexes 18-20) and mean image dimension (indexes 24-26) meta-features peak higher and appear more promising.

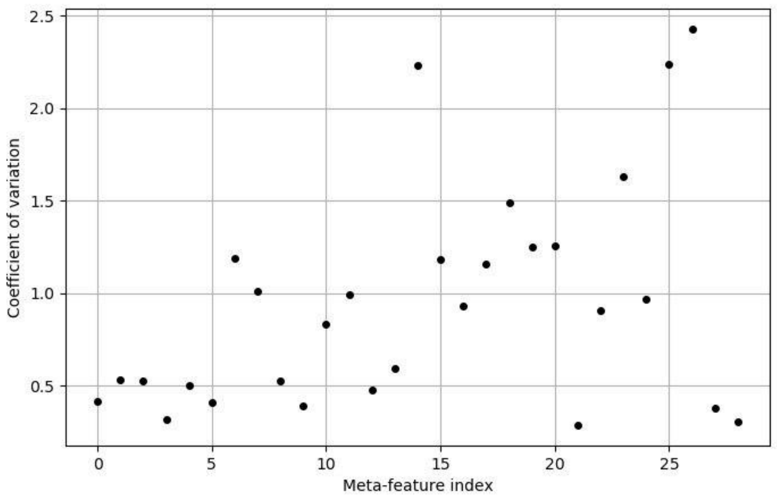


Figure 5.4: Coefficient of variation of meta-features across all datasets

5.4 Computation time

ConvNet training accounts for a massive portion of overall computation time whereas meta-feature extraction and meta-model training take insignificant times when compared. Therefore, two systems were used for different purposes:

- The meta-feature extraction and meta-learner processes were ran on a machine with an AMD Ryzen 5 3600X 6-Core CPU and Nvidia GeForce RTX 2060 SUPER GPU. Due to power consumption concerns and it being a personal computer, extensive machine learning tasks could not be performed.
- The deep learning part of the project was ran on a HPC [1] provided by Fraunhofer AICOS Portugal. It runs on a cluster of Tesla V100-PCIE-16GB GPUs.

The computation times for all these tasks can be further analysed in Appendix A.

[1] High-performance Computing. Generally referring to the concept of aggregating multiple computers in a way that delivers high performance for engineering, science or business tasks.

Chapter 6

Results

6.1 Exploratory analysis of metadata

In order to exactly measure how much dropout affects performance, we have plotted the difference between the predictive accuracy of a basic network and of a dropout-enabled of this network. A positive value means that dropout effectively boosted the convolutional network model. Negative values illustrate cases where applying dropout is accompanied with a decrease in efficency. These results can be found in Figure 6.1.

We can observe a roughly even distribution for some instances where dropout strongly hinders predictive performance and others where it firmly improves it. Due to using the same base network to all datasets, in most cases its size is not entirely appropriate for the dataset in question. Whenever a network is too large for the dataset, the model learns the detail and noise of the training set, causing overfit. In these cases, regularization is welcome and should be illustrated by the positive values on the plot. However, if the network is too simplistic, it might already be struggling to make accurate predictions. Here, inducing dropout will often cause further complications because it is removing the bare minimum predictive units, causing negative values on the plot.

The *malaria*, *uc_merced* datasets and *daimlerpedcls* datasets – with respective predictive differences of -0.19, -0.15 and -0.09 – stand as the ones most negatively affected by dropout. In an effort to understand why this is, let us analyse their specifications to draw possible conclusions:

- The *malaria* dataset hosts over 27558 images of segmented cells from a thin blood smear on a glass slide with equal instances of parasitized and uninfected cells [25]. It is therefore

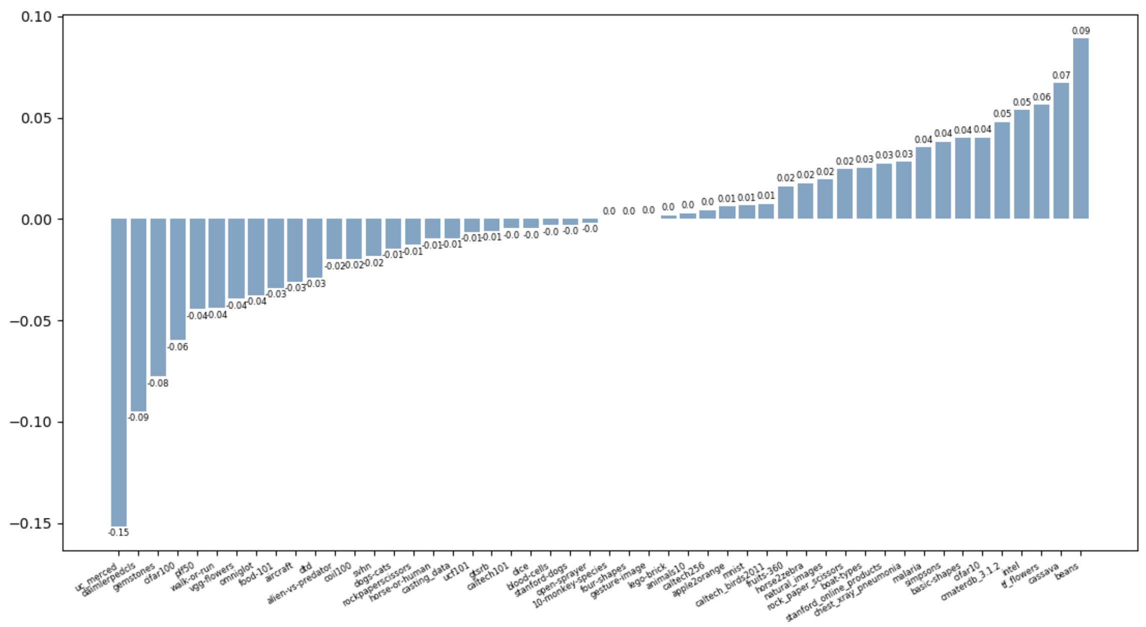


Figure 6.1: Accuracy distance between $p = 0.5$ dropout-trained networks and basic networks across the meta-database.

categorized as a binary classification dataset. Figure 6.2 illustrates a small sample of images found within the dataset. Interestingly, another study that trains on this exact dataset uses a similar architecture, even also employing dropout $p = 0.5$ to the outputs of the first fully-connected layer [40]. Their proposed model on cell level imagery achieved an accuracy value of 0.986 with dropout when ours netted 0.91 without it, which is impressive for a network that has not been fine-tuned to fit the dataset. When we activated the dropout layer, the score plummeted to an average of 0.72. This divergence is most likely due to the fact that authors went with a step of hyperparameter optimization that was impossible for us and usage of a different optimizer – SGD while we used Adam on our models. The fact that the base network did not benefit from dropout appears to be an indicative that it was not overfitting to begin with and its original size fit the dataset well.

- The *uc_merced* dataset is described as a 21-class land use image dataset extracted from the USGS National Map Urban Area Imagery collection for various urban areas around the country. Despite featuring only 100 images per class, it still managed to achieve 0.36 accuracy.

- The Daimler Pedestrian Segmentation Benchmark dataset (dubbed *daimlerpedcls* in our meta-database) consists of a collection of pedestrian and non-pedestrian images. It suffered the third biggest cut in accuracy when dropout was switched on, dropping from 0.96 to 0.87.

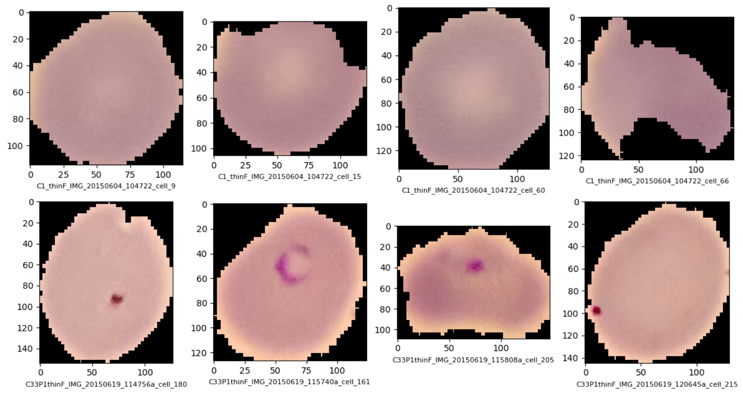

C1_thinF_IMG_20150604_104722_cell_9
C1_thinF_IMG_20150604_104722_cell_15
C1_thinF_IMG_20150604_104722_cell_60
C1_thinF_IMG_20150604_104722_cell_66
C33P1thinF_IMG_20150619_114756a_cell_180
C33P1thinF_IMG_20150619_115740a_cell_161
C33P1thinF_IMG_20150619_115808a_cell_205
C33P1thinF_IMG_20150619_120645a_cell_215

Figure 6.2: Sample from the *malaria* dataset. Images on the top row belong to uninfected class, while images on the bottom row belong to the parasitized class.

The *beans*, *cassava* and *tf_flowers* datasets were the most positively affected by dropout, with respective differences of +0.09, +0.07 and +0.06.

- The *beans* dataset is comprised of bean images taken in the field using smartphone cameras. It consists of 3 classes: 2 disease classes – Angular Leaf Spot and Bean Rust – and an healthy class. A sample from the dataset was made available on Figure 6.3. Applying dropout boosted its original performance of 0.69 to 0.78, suggesting that the default network was too complex for the dataset and simplifying it resulted on significantly better predictive power.

- Similarly to *beans*, *cassava* also consists of leaf images depicting four disease conditions and a healthy class, with a total of 9430 labelled images. This dataset is unbalanced with two of the classes amounting to 72% of the images.

- Finally, *tf_flowers* includes flower images from 5 different classes – daisy, dandelion, roses, sunflowers and tulips. There is a total of 3670 images distributed unevenly across the classes.

Interestingly, the top 3 datasets most positively affected by dropout relate to plants with the first two being quite similar in premise and image content. Whether this is a coincidence or an indication that image content and dataset morphology are directly related to the effects of dropout we could not determine.

In many other datasets, regularizing the network yielded practically zero benefits. This is easily verified on the *10-monkey-species*, *animals10*, *four-shapes* and *lego-brick* datasets, where accuracy only varied by less than ± 0.01.

Through empirical analysis of these datasets, it is difficult to determine the model's behaviour when dropout switches on and off. Naturally, this difficulty should be linked to the near-impossible

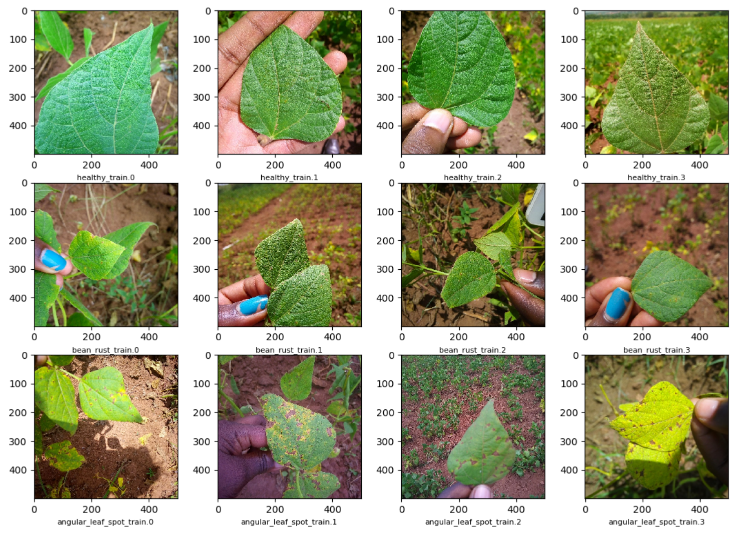


Figure 6.3: Sample from the *beans* dataset. *healthy* class on the top row; *bean_rust* class on the middle row; and *angular_leaf_spot* on the bottom row.

human interpretation of deep nets reasoning, so we should not be expecting to draw many conclusions from empirical analysis of the datasets' performance.

6.2 Regarding predicting the effect of dropout (RQ1)

For statistical meta-features, in order to measure the dataset's predictive power, we have run each through three different classifiers: (1) *LogisticRegression* from *scikit-learn* with L2 norm penalty, $c = 0.01$ and a maximum of 5000 iterations; (2) a default *RandomForestClassifier* also from *scikit-learn* and (3) a default binary logistic regression from XGBoost's *XGBClassifier*.

Meta-dataset	**LogisticRegression**	**RandomForestClassifier**	**XGBClassifier**
1	0.48 ± **0.193**	0.463 ± 0.206	0.443 ± 0.195
2	**0.508** ± 0.202	0.445 ± **0.193**	**0.449** ± 0.197
3	0.478 ± 0.204	0.446 ± 0.198	0.417 ± **0.183**
4	0.507 ± 0.223	**0.435** ± 0.22	0.405 ± 0.201

Table 6.1: Predictive power of different classifiers applied to each statistical meta-dataset

The results illustrated on Table 6.1 are the averaged means and standard deviations of applying 10-fold cross-validation 10 times. Based on the results, we may derive two main assumptions: simpler classifiers generally appear to yield better results than tree-based classifiers. Yet still, predictions seem random. Even if the second meta-dataset consistently averages over 50% accuracy, it is probably not statistically significant. This appears to support the idea that the selected meta-features do not contain information useful enough to predict the effect of dropout in general classification problems.

Therefore, we resorted to experimenting with deep features, across three different encoders: VGG16, VGG19 and MobileNetV1. Due to the features' massive size (MobileNetV1's filter count of 1024 results on over 16 million features), we trimmed each dataset's feature array to varying lengths. The ResNet50 model had to be dropped because its size, which doubled MobileNetV1 and quadrupled VGG16 and VGG19, could not fit in memory. For this experiment, we resorted to logistic regression only, because of its optimistic results on the statistical features, keeping the 10-fold cross validation.

Trim length	**VGG16**	**VGG19**	**MobileNetV1**
10^2	0.465 ± **0.174**	0.491 ± **0.182**	0.387 ± **0.184**
10^3	**0.478** ± 0.206	0.547 ± 0.194	0.449 ± 0.194
10^4	0.461 ± 0.196	**0.564** ± 0.214	**0.516** ± 0.2

Table 6.2: Predictive power of logistic regression for deep features from distinct models, according to varying sample sizes

Once again, the shown results in Table 6.2 are averaged from 10 repetitions. For VGG16, regardless of trim length, accuracy is certainly low. For the VGG19 variant, accuracy was surprisingly high, reaching an average score of 0.564 for the highest trim length. However, due to its also high variance, it probably still equals to random guessing. In MobileNetV1's case, the model only broke the 50% threshold for a trim length of 10^4. Generally, keeping features as opposed to dropping them appears to generate better results, with exception to VGG16 where predictions remained stable.

Overall, these results appear to support the idea that the interpretation of pre-trained networks provide more meaningful information compared to the selected statistical ones. Unfortunately, because of the networks' abstract interpretation of the datasets, undergoing any sort of feature importance step would be pointless. So even if VGG19's analysis of the meta-dataset does appear promising, we are still unable to comprehend as to why it did so well compared to other methods and which patterns it caught up.

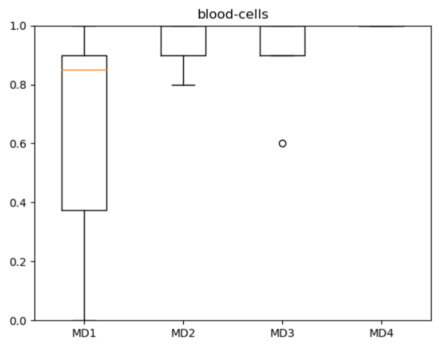
6.3 Regarding sampling approaches (RQ2, RQ3)

As described in subsection 4.2.1, we generated four meta-datasets, where the last two hoped to reduce variation of the results. We registered, across 10 repetitions of the 10-fold cross-validation step, how many times a dataset had been correctly classified. We then computed the mean and standard deviation of correct prediction rates across all samples. Due to the fact that the fourth meta-dataset is, by design, only comprised of one sample, standard deviation could not be calculated.

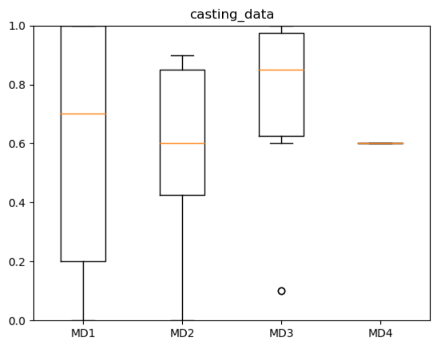
Meta-dataset	R_1	R_2	R_3	**Total**
MD_1	0.477 ± 0.27	0.474 ± 0.268	0.475 ± 0.271	0.475 ± 0.27
MD_2	0.51 ± 0.264	0.512 ± 0.258	0.515 ± 0.264	$\mathbf{0.512} \pm 0.262$
MD_3	0.481 ± 0.244	0.476 ± 0.243	0.474 ± 0.243	$0.477 \pm \mathbf{0.243}$
MD_4	0.494 ± 0.00	0.496 ± 0.00	0.51 ± 0.00	0.5 ± 0.00

Table 6.3: Mean and standard deviation of correct prediction rates across each dataset's samples

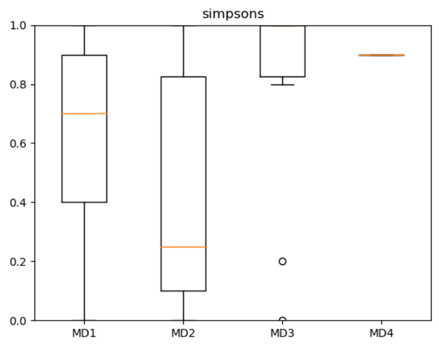
Table 6.3 clearly illustrates that an increase in the meta-dataset index is proportional to prediction stability. In other words, in the third meta-dataset, across the all samples, predictions tended to be more stable than on other approaches.



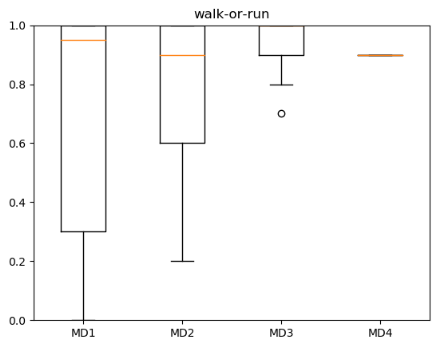
(a) Boxplot of the *blood-cells* dataset



(b) Boxplot of the *casting_data* dataset



(c) Boxplot of the *simpsons* dataset

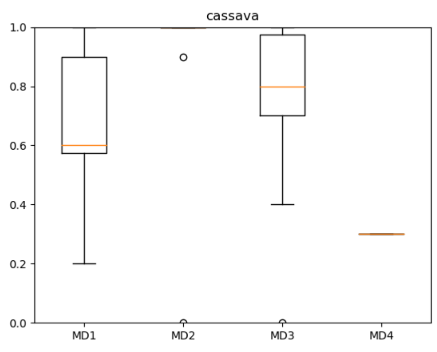


(d) Boxplot of the *walk-or-run* dataset

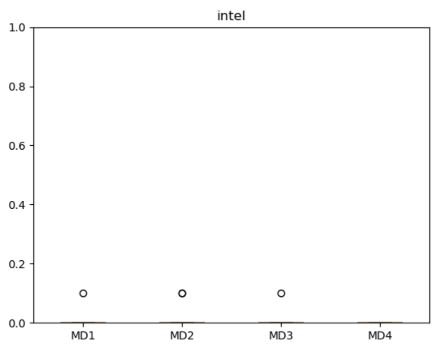
Figure 6.4: Boxplots from datasets that support increasing stability

Performing boxplots on top of this data provided us with a better visual understanding of error and outliers. Figure 6.4 illustrates a set of examples where *MD3* shows more consistent values and should make predictions more dependable than on the other meta-datasets.

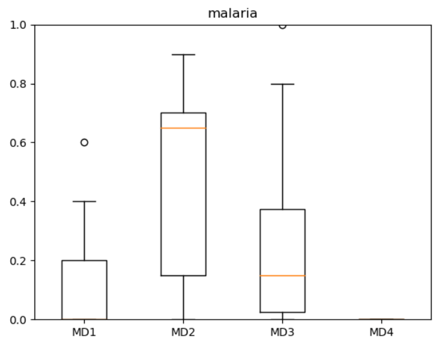
Although this is the norm (proven by the results on Table 6.3), observations can vary greatly. For instance, in subplot 6.5a, on $MD2$, *cassava* has been correctly predicted 100% of the times with the exception of an outlier. $MD3$ did slightly help reducing $MD1$'s variance but is well worse compared to $MD2$. For the *intel* dataset, on subplot 6.5b, all meta-datasets performed equally poorly, showing that the inclusion of *intel* is hindering the model's predictive power regardless of the data's structure. On the last two subplots we notice that *MD*1 for the *malaria* dataset (in subplot 6.5c) and *MD*2 for *stanford_online_products* (in subplot 6.5d) respectively outperformed *MD*3 in terms of both accuracy and standard deviation.



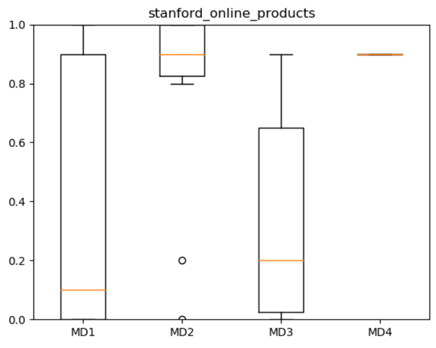
(a) Boxplot of the *cassava* dataset



(b) Boxplot of the *intel* dataset



(c) Boxplot of the *malaria* dataset



(d) Boxplot of the *stanford_online_products* dataset

Figure 6.5: Boxplots from datasets that do not support increasing stability

Overall, even if this analysis does show a tendency for *MD*3 structuring to yield more stable results, they should be carefully interpreted on a case-to-case basis. A complete list of boxplots for all datasets was made available on Appendix A.1, A.2 and A.3.

6.4 Answering research questions

With the context of the results, we now take time to directly answer the research questions established in Chapter 4 that were indirectly answered throughout the document, in a way to summarize this chapter.

- **RQ1**: *Can a meta-learning approach be used to predict the effect of dropout?*
  The meta-model only managed to consistently score above 50% in the case of VGG19 deep features at 10^4 trim size and on the 2nd statistical meta-dataset. However, we still consider these results too variable to extract any meaningful information. Due to the fact that

the highest scoring features were the product of a pre-trained VGG19 network, analysing its interpretation is too ambitious. If advancements want to be accomplished in the area of network explainability, a better set of human-interpretable meta-features that accurately describe the datasets must first be assessed.

- **RQ2**: *Does the sampling approach proposed by Sonsbeek, 2019 induce variance in the data?*
  We were able to observe that meta-dataset *MD*1 consistently generates predictions with high variance while *MD*3 yielded similar accuracy rates but with noticeably lower variance. Therefore, it does appear that the sampling approach proposed by the previous work is a source of variance when compared to a more careful approach like ours, even if the model's predictive power stays the same.

- **RQ3**: *Does the aggregation of features obtained on samples using average and standard deviation reduce the effect of variance?*
  We were able to find meaningful decreases in variance by computing the mean and standard deviation of samples instead of keeping them as they were. However, the meta-datasets still experience less than ideal variance. Although it does look like a step forward, future work should further experiment with other sampling calculations.

6.5 Computation time

The workflow of gathering and preparing a dataset, extracting its meta-features and measuring its performance across different dropout rates is very computationally taxing.

Regarding statistical meta-feature extraction, the runtime fluctuated depending on whether, in a dataset, all images have the same dimensions. Not having so prompts the process to extract mutual information meta-features which can take hours to gather. These meta-features analyze image similarity by zooming and overlaying two data samples and performing calculations on top of a bi-dimensional histogram of ravelled images. The procedure of overlaying multiple samples through the zooming operation accounts for most of the computation time.

Perhaps counter-intuitively, extracting deep learning meta-features for the entire database consistently took only a portion of time used by the statistical approach, across all encoders. This is because all the models utilized were pre-trained and the networks only had to make a prediction.

A full list of recorded computation times for all methods are available in Appendix A.

While statistical meta-feature extraction for all datasets took only a few hours and could be ran locally, training several networks for multiple datasets ended up only being viable in a cluster.

Chapter 7

Conclusions and Future Work

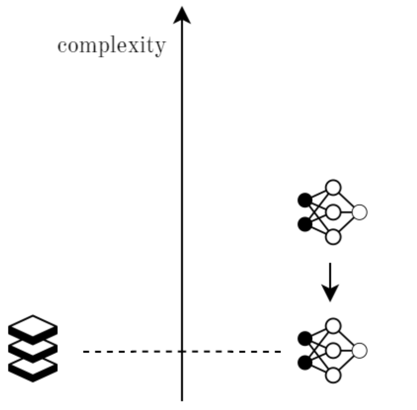
7.1 Discussion

During this dissertation, dropout has been applied only to the fully-connected layers of convolutional neural networks. Yet dropout has also been applied to convolutional layers albeit with mixed results. In some cases, it appears applying standard dropout before 1x1 convolutions generally increases training time but does not prevent overfitting [49]. But in others, it actually produces better results. For instance, multiplying Bernoulli noise into the feature map has been proven itself as a solid way to regularize networks [18, 29]. However, these dropout mechanisms are fundamentally different. Therefore, this dissertation's investigation should only be valid in the domain of dropout applied to fully-connected layers.

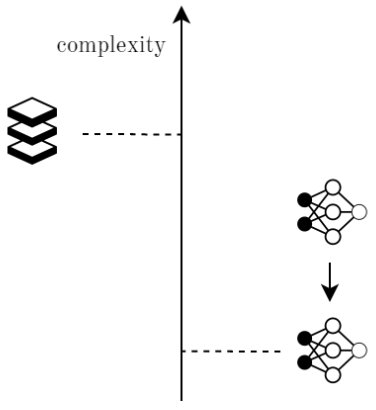
We believe one of the main reasons that directly affected the model's performance is the fact that perhaps the selected features do not represent complexity well.

In our experimental setup we have limited our ConvNet's architecture to two versions – one with and other without dropout – and applied it to all datasets. In Figure 7.1a, the ideal case for dropout regularization is illustrated: the data is too simple for the network and thus the model overfits. Employing dropout balances complexities, closing its gap and resulting on better predictive power. In Figure 7.1b the opposite happens. Regularizing an otherwise already too simple network could make its accuracy drop further down because now the complexity distance between dataset and network is larger than ever.

So our hypothesis is that the distance between dataset and model complexity is directly tied to the effects of dropout. Provided that we make the only moving variable the datasets, then it becomes a matter of extracting features that represent whether a dataset is simple or complex. However, exactly gauging what makes a dataset complex for a deep model is by all means not an easy task. Even if such investigation does not land within reach of this dissertation's focus,



(a) Applying dropout to an overfit network



(b) Applying dropout to an underfit network

Figure 7.1: Comparison of applying dropout to networks and datasets with different complexities

we do wonder whether our only image-based statistical features are enough and whether adding morphological characteristics of a dataset – e.g. number of classes, class imbalance, number of examples – would have been more representative of this concept.

This hypothesis is closely tied to the findings of how much the sampling approaches influence data variance. While we were capable to find that, generally, aggregating samples and computing new features on top of it appears to reduce variance, we were unable to improve accuracy. More adequate features should provide us with a better understanding of the actual implications of the sampling approach not only for variance but also for a model's predictive power.

7.2 Future work

Due to the fact that the type of selected statistical features were uncapable of providing meaningful results, we urge future work to experiment with other statistical features that are not solely image-based but also represent the dataset's morphological characteristics. Regarding deep features, even if the network interpretation of pre-trained models provided the most promising results, its predictions lack interpretability and do not exactly provide useful insights as to why dropout was powerful or not in different scenarios. So while said features are favorable for understanding whether dropout can be modeled with meta-learning, the explainability portion would lack.

Experimenting with other convolutional neural network architectures instead of only using a single one is likely to provide a baseline as to how much the density of a network influences the predictive confidence of the meta-model.

Additionally, regarding the variance issue, it is likely that simply computing the average and standard deviation for the aggregated samples is not enough to achieve significantly superior results. We encourage future work on this feature engineering step to consider other statistical metrics such as kurtosis, skew, coefficient of variance, and others.

Appendix A

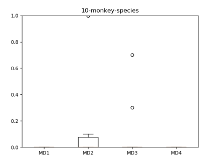
Meta-database

Dataset	**class_no**	**train_size**	**test_size**	**val_size**	**total_size**	**unique_shapes**
10-monkey-species	10	983	115	272	1370	True
aircraft	100	2934	400	3333	6667	True
alien-vs-predator	2	624	70	200	894	True
animals10	10	21197	2623	2359	26179	True
apple2orange	2	1812	514	202	2528	False
basic-shapes	3	243	30	27	300	False
beans	3	1034	128	133	1295	False
blood-cells	4	8960	2487	997	12444	False
boat-types	9	1177	150	135	1462	True
caltech101	102	5784	958	772	7514	True
caltech256	257	19402	3180	2548	25130	True
caltech_birds2011	200	7445	1191	990	9626	True
cassava	5	5656	1885	1889	9430	True
casting_data	2	5969	715	664	7348	False
chest_xray_pneumonia	2	5216	624	16	5856	True
cifar10	10	45000	10000	5000	60000	False
cifar100	100	36000	4000	10000	50000	False
cmaterdb_3.1.2	50	10800	3001	1200	15001	True
coil100	100	4500	800	600	5900	False
daimlerpedcls	2	21168	2352	5880	29400	False
dice	6	12852	1432	2102	16386	False
dogs-cats	2	7200	2000	800	10000	True
dtd	47	1692	188	1880	3760	True
food-101	101	81810	10100	9090	101000	True
four-shapes	4	12124	1498	1348	14970	False
fruits-360	120	54405	20622	6093	81120	False
gemstones	87	2529	363	327	3219	True
gesture-image	37	44955	5550	4995	55500	False
gtsrb	43	28214	3153	7842	39209	True
horse-or-human	2	924	103	256	1283	False
horse2zebra	2	2160	260	241	2661	False
intel	6	12626	3000	1408	17034	False
lego-brick	16	5166	638	575	6379	False
malaria	2	17636	2756	2206	22598	True
mnist	10	26873	4206	3362	34441	False
natural_images	8	5582	693	624	6899	True
omniglot	1623	14607	3246	6492	24345	False
open-sprayer	2	5423	604	670	6697	False
plf50	18	450	72	54	576	False
rockpaperscissors	3	1770	220	198	2188	False
rock_paper_scissors	3	2268	372	252	2892	False
simpsons	42	16918	2113	1902	20933	True
stanford-dogs	120	16567	2110	1903	20580	True
stanford_online_products	12	97144	12000	10800	119944	True
svhn	10	42489	4728	26040	73257	False
tf_flowers	5	2970	369	331	3670	True
ucf101	101	6780	805	1952	9537	False
uc_merced	21	1701	210	189	2100	True
vgg-flowers	102	918	102	1020	2040	True
walk-or-run	2	539	141	61	741	False

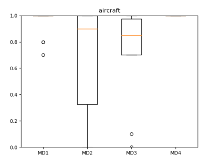
Table A.1: Morphological characteristics of the meta-database

Dataset	**STAT**	**VGG16**	**VGG19**	**MOBILENETV1**	**RESNET-50**
10-monkey-species	3629.19	10.97	7.49	8.18	12.03
aircraft	24.19	5.80	5.35	6.54	10.29
alien-vs-predator	216.30	5.74	4.68	5.14	8.14
animals10	324.22	9.83	10.53	11.55	14.67
apple2orange	9.22	6.26	5.67	6.33	9.91
basic-shapes	1.17	3.48	2.72	4.07	8.14
beans	18.52	6.24	5.49	6.22	9.31
blood-cells	12.77	8.09	7.76	11.22	12.49
boat-types	3064.37	7.55	6.67	7.35	10.69
caltech101	292.00	6.78	7.07	7.77	11.42
caltech256	611.37	8.76	8.73	10.64	13.46
caltech_birds2011	604.18	8.14	7.88	8.70	14.21
cassava	1061.03	7.61	7.74	8.54	12.15
casting_data	12.58	7.91	7.74	8.37	11.93
chest_xray_pneumonia	5757.74	9.29	9.22	9.70	13.57
cifar10	41.75	8.81	9.16	10.33	14.46
cifar100	15.28	9.38	9.57	10.26	14.00
cmaterdb_3.1.2	30.60	7.98	7.93	9.01	12.23
coil100	7.61	6.04	5.96	7.06	10.64
daimlerpedcls	38.13	9.01	8.58	9.60	13.93
dice	21.74	8.92	8.19	9.10	13.04
dogs-cats	637.27	8.41	8.49	9.04	12.96
dtd	29.66	4.96	4.87	6.00	9.02
food-101	390.89	11.55	11.64	13.42	16.63
four-shapes	11.36	10.39	9.82	10.30	15.06
fruits-360	22.75	17.13	10.18	12.03	15.50
gemstones	515.28	10.03	7.97	8.34	11.62
gesture-image	21.63	20.59	11.69	12.55	16.43
gtsrb	36.05	14.51	8.90	10.25	13.91
horse-or-human	10.09	8.03	5.85	6.41	9.70
horse2zebra	9.87	7.59	7.16	7.40	10.67
intel	12.34	10.33	9.53	10.19	14.32
lego-brick	8.84	9.12	8.02	8.97	12.33
malaria	110.11	11.81	8.25	10.03	12.92
mnist	8.12	11.51	6.16	7.96	11.16
natural_images	190.00	9.94	9.18	10.35	13.73
omniglot	14.30	11.51	8.35	9.41	13.09
open-sprayer	13.42	9.91	8.07	8.79	12.67
plf50	3.92	8.95	4.00	4.64	8.56
rockpaperscissors	8.97	6.84	6.24	6.72	10.10
rock_paper_scissors	8.46	6.99	6.75	7.01	10.50
simpsons	732.43	12.84	9.46	10.32	14.33
stanford-dogs	661.70	13.07	9.68	10.81	15.15
stanford_online_products	674.56	50.43	12.22	13.34	17.16
svhn	41.77	18.43	10.43	11.45	15.39
tf_flowers	391.63	8.85	7.65	8.61	12.07
ucf101	11.27	8.95	8.53	10.33	12.62
uc_merced	19.89	8.12	6.63	7.07	10.59
vgg-flowers	31.41	6.94	5.35	5.36	9.40
walk-or-run	9.37	6.28	4.56	5.17	8.30
Total	20431.32	506.60	389.76	437.95	616.60

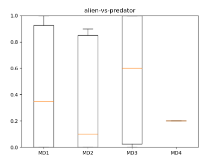
Table A.2: *Machine A*'s compute time (in seconds) for meta-feature extraction across various techniques



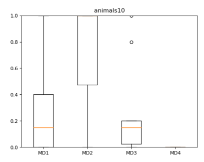
(a) Boxplot of the *10-monkey-species* dataset



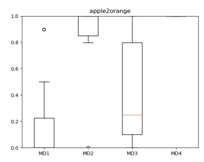
(b) Boxplot of the *aircraft* dataset



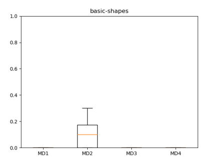
(c) Boxplot of the *alien-vs-predator* dataset



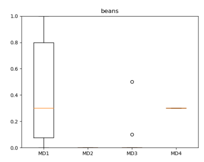
(d) Boxplot of the *animals10* dataset



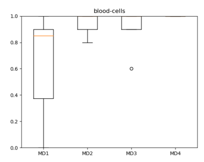
(e) Boxplot of the *apple2orange* dataset



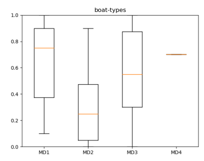
(f) Boxplot of the *basic-shapes* dataset



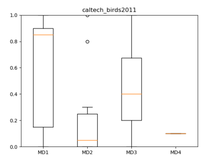
(g) Boxplot of the *beans* dataset



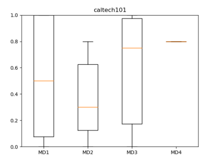
(h) Boxplot of the *blood-cells* dataset



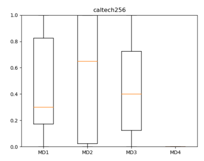
(i) Boxplot of the *boat-types* dataset



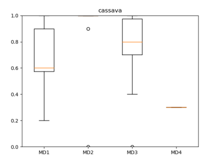
(j) Boxplot of the *caltech_birds2011* dataset



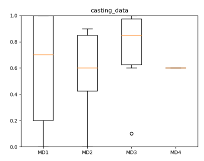
(k) Boxplot of the *caltech101* dataset



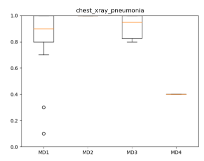
(l) Boxplot of the *caltech256* dataset



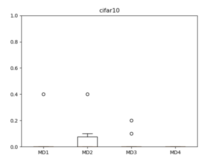
(m) Boxplot of the *cassava* dataset



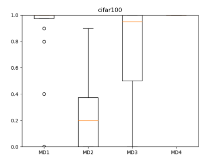
(n) Boxplot of the *casting_data* dataset



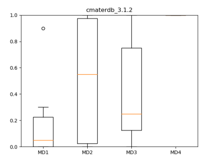
(o) Boxplot of the *chest_xray_pneumonia* dataset



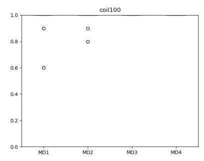
(p) Boxplot of the *cifar10* dataset



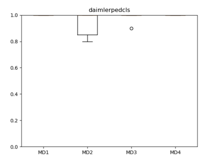
(q) Boxplot of the *cifar100* dataset



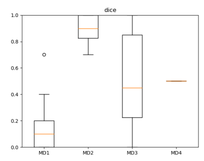
(r) Boxplot of the *cmaterdb_3.1.2* dataset



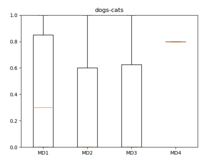
(s) Boxplot of the *coil100* dataset



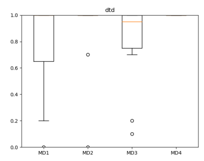
(t) Boxplot of the *daimlerpedcls* dataset



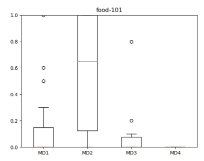
(u) Boxplot of the *dice* dataset



(v) Boxplot of the *dogs-cats* dataset

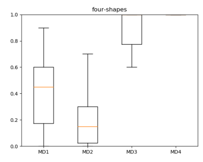


(w) Boxplot of the *dtd* dataset

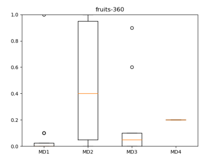


(x) Boxplot of the *food-101* dataset

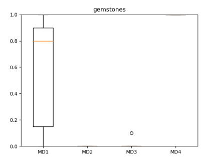
Figure A.1: Set #1 of dataset boxplots regarding prediction stability



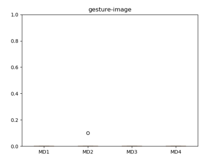
(a) Boxplot of the *four-shapes* dataset



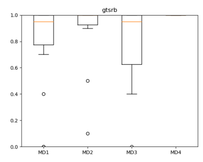
(b) Boxplot of the *fruits-360* dataset



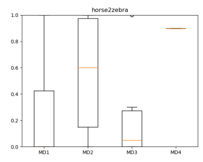
(c) Boxplot of the *gemstones* dataset



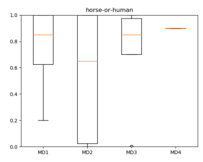
(d) Boxplot of the *gesture-image* dataset



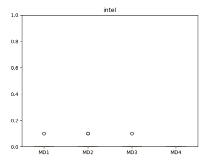
(e) Boxplot of the *gt-srb* dataset



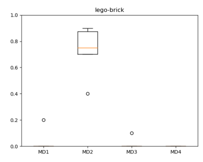
(f) Boxplot of the *horse2zebra* dataset



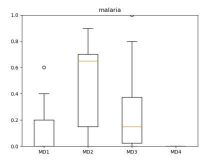
(g) Boxplot of the *horse-or-human* dataset



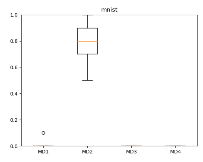
(h) Boxplot of the *in-tel* dataset



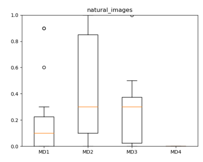
(i) Boxplot of the *lego-brick* dataset



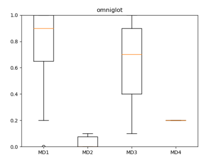
(j) Boxplot of the *malaria* dataset



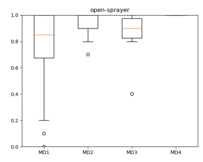
(k) Boxplot of the *mnist* dataset



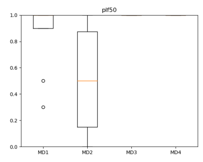
(l) Boxplot of the *natural_images* dataset



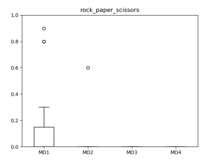
(m) Boxplot of the *omniglot* dataset



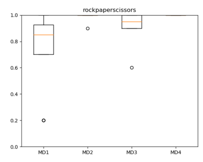
(n) Boxplot of the *open-sprayer* dataset



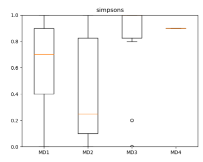
(o) Boxplot of the *plf50* dataset



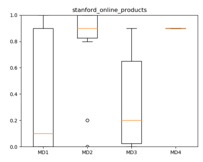
(p) Boxplot of the *rock_paper_scissors* dataset



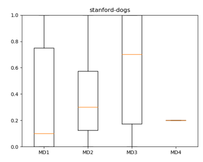
(q) Boxplot of the *rockpaperscissors* dataset



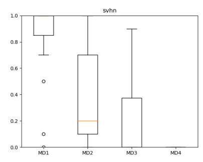
(r) Boxplot of the *simpsons* dataset



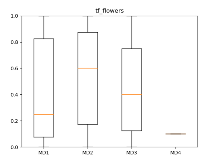
(s) Boxplot of the *stan-ford_online_products* dataset



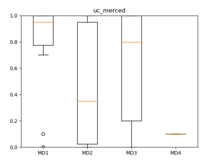
(t) Boxplot of the *stanford-dogs* dataset



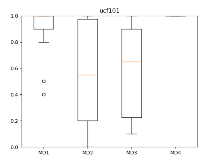
(u) Boxplot of the *svhn* dataset



(v) Boxplot of the *tf_flowers* dataset

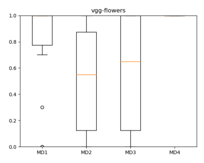


(w) Boxplot of the *uc_merced* dataset

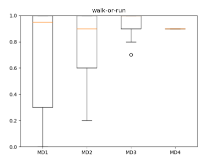


(x) Boxplot of the *ucf101* dataset

Figure A.2: Set #2 of dataset boxplots regarding prediction stability



(a) Boxplot of the *vgg-flowers* dataset



(b) Boxplot of the *walk-or-run* dataset

Figure A.3: Set #3 of dataset boxplots regarding prediction stability

Bibliography

[1] The 5 classification evaluation metrics every data scientist must know. https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226. Accessed: 2020-02-06.

[2] Cs231n convolutional neural networks for visual recognition. https://cs231n.github.io/convolutional-networks. Accessed: 2020-02-06.

[3] mfe: Meta-feature extractor - cran. https://cran.r-project.org/web/packages/mfe/vignettes/mfe-vignette.html. Accessed: 2020-02-07.

[4] Multilayer perceptron example. https://github.com/rcassani/mlp-example. Accessed: 2020-06-27.

[5] The mostly complete chart of neural networks, explained. https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explaine Accessed: 2020-06-27.

[6] Convolutional neural network (cnn) tutorial. https://www.tensorflow.org/tutorials/images/cnn. Accessed: 2020-06-24.

[7] An intuitive explanation of convolutional neural networks. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. Accessed: 2020-02-06.

[8] Machine learning challenge winning solutions. https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions, 2020.

[9] Suresh Chandra Satapathy Anand J. Kulkarni. *Optimization in Machine Learning and Applications*. Springer, 2020.

[10] Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3084–3092. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5032-adaptive-dropout-for-training-deep-neural-networks.pdf.

[11] Anup Satish Balte, Nitin Namdeo Pise, and Parag Kulkarni. Meta-learning with landmarking: A survey. 2014.

[12] P Brazdil, C Giraud-Carrier, C Soares, and R Vilalta. *Metalearning*. Cognitive Technologies. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-73262-4. doi: 10.1007/978-3-540-73263-1. URL http://link.springer.com/10.1007/978-3-540-73263-1.

[13] G. F. C. Campos, S. Barbon, and R. G. Mantovani. A meta-learning approach for recommendation of image segmentation algorithms. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 370–377, 2016.

[14] Ciro Castiello, Giovanna Castellano, and Anna Fanelli. Meta-data: Characterization of input features for meta-learning. pages 457–468, 07 2005. doi: 10.1007/11526018_45.

[15] Sanjiv Ranjan Das. Data science: Theories, models, algorithms, and analytics, 2017.

[16] Luke Davis, Barry-John Theobald, and Anthony Bagnall. Automated bone age assessment using feature extraction. 08 2012. doi: 10.1007/978-3-642-32639-4_6.

[17] Keith D. Foote. A brief history of deep learning. https://www.dataversity.net/brief-history-deep-learning/#:~:text=The%20history%20of%20Deep%20Learning,to%20mimic%20the%20thought%20process. Accessed: 2020-07-21.

[18] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference, 2015.

[19] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[20] Thomas Hartmann. Meta-modelling meta-learning. https://medium.com/datathings/meta-modelling-meta-learning-34734cd7451b. Accessed: 2020-07-21.

[21] Saihui Hou and Zilei Wang. Weighted channel dropout for regularization of deep convolutional neural network. In *AAAI*, 2019.

[22] Jabbar Salman Hussain. Deep learning black box problem. 2019.

[23] Frank Hutter, Jörg Lücke, and Lars Schmidt-Thieme. Beyond manual tuning of hyperparameters. *KI - Künstliche Intelligenz*, 29, 07 2015. doi: 10.1007/s13218-015-0381-0.

[24] Yoshua Bengio Ian Goodfellow and Aaron Courville. *Deep learning*. MIT Press, 2016. ISBN 9780262035613.

[25] Stefan Jaeger. Malaria datasets. https://lhncbc.nlm.nih.gov/publication/pub9932. Accessed: 2020-06-30.

[26] A. K. Jain, Jianchang Mao, and K. M. Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3):31–44, March 1996. ISSN 1558-0814. doi: 10.1109/2.485891.

[27] Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *CoRR*, abs/1602.02410, 2016. URL http://arxiv.org/abs/1602.02410.

[28] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[29] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding, 2015.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[32] Hyunkwang Lee, Sehyo Yune, Mohammad Mansouri, Myeongchan Kim, Shahein H. Tajmir, Claude Emmanuel Guerrier, Sarah A Ebert, Stuart R. Pomerantz, Javier M. Romero, Shahmir Kamalian, Ramón González González, Michael H. Lev, and Synho Do. An explainable deep-learning algorithm for the detection of acute intracranial haemorrhage from small datasets. *Nature Biomedical Engineering*, 3:173–182, 2018.

[33] Sokbae Lee and Serena Ng. An econometric perspective on algorithmic subsampling. *arXiv: Econometrics*, 2020.

[34] Xuan Liu, Xiaoguang Wang, and Stan Matwin. Interpretable deep convolutional neural networks via meta-learning. *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2018.

[35] Matilda Lorentzon. Feature extraction for image selection using machine learning, 2017.

[36] Asma Naseer and Kashif Zafar. Comparative analysis of raw images and meta feature based urdu ocr using cnn and lstm. *International Journal of Advanced Computer Science and Applications*, 9, 01 2018. doi: 10.14569/IJACSA.2018.090157.

[37] Szilard Pafka. benchm-ml. https://github.com/szilard/benchm-ml, 2019. URL https://github.com/szilard/benchm-ml.

[38] Yonghong Peng, Peter Flach, Carlos Soares, and Pavel Brazdil. Improved dataset characterisation for meta-learning. volume 2534, pages 141–152, 11 2002. doi: 10.1007/3-540-36182-0_14.

[39] Peter Sadowski Pierre Baldi. Understanding dropout, 2013.

[40] Sivaramakrishnan Rajaraman, Sameer K. Antani, Mahdieh Poostchi, Kamolrat Silamut, Md. Ali Hossain, Richard James Maude, Stefan Jaeger, and George R. Thoma. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ*, 6, 2018.

[41] Nina Schaaf and Marco F. Huber. Enhancing decision tree based interpretation of deep neural networks through l1-orthogonal regularization. *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 42–49, 2019.

[42] Kevin Schawinski, Ce Zhang, Hantian Zhang, Lucas Fowler, and Gokula Krishnan Santhanam. Generative adversarial networks recover features in astrophysical images of galaxies beyond the deconvolution limit. *Monthly Notices of the Royal Astronomical Society: Letters*, 467(1):L110–L114, 01 2017. ISSN 1745-3925. doi: 10.1093/mnrasl/slx008. URL https://doi.org/10.1093/mnrasl/slx008.

[43] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2014.09.003. URL http://www.sciencedirect.com/science/article/pii/S0893608014002135.

[44] Saddys Segrera, Joel Pinho, and María N. Moreno. Information-theoretic measures for meta-learning. In Emilio Corchado, Ajith Abraham, and Witold Pedrycz, editors, *Hybrid Artificial Intelligence Systems*, pages 458–465, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-87656-4.

[45] Thomas Grant Slatton. A comparison of dropout and weight decay for regularizing deep neural networks, 2014.

[46] Tom Van Sonsbeek. Meta-learning for medical image segmentation, 2019.

[47] Nitish Srivastava. Improving neural networks with dropout, 2013.

[48] Ahmad S. Tarawneh, Dmitry Chetverikov, Chaman Singh Verma, and Ahmad B. A. Hassanat. Stability and reduction of statistical features for image classification and retrieval: Preliminary results. *2018 9th International Conference on Information and Communication Systems (ICICS)*, pages 117–121, 2018.

[49] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. Efficient object localization using convolutional networks, 2014.

[50] Li Wan, Matthew Zeiler, Sixn Zhang, Yann Lecun, and Rob Fergus. Regularization of neural networks using dropconnect. 01 2013.

[51] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. ESRGAN: enhanced super-resolution generative adversarial networks. *CoRR*, abs/1809.00219, 2018. URL http://arxiv.org/abs/1809.00219.

[52] Haibing Wu and Xiaodong Gu. Towards dropout training for convolutional neural networks. *Neural Networks*, 71:1 – 10, 2015. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2015.07.007. URL http://www.sciencedirect.com/science/article/pii/S0893608015001446.

[53] Matthew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks, 2013.

[54] Yu-Dong Zhang, Chichun Pan, Junding Sun, and Chaosheng Tang. Multiple sclerosis identification by convolutional neural network with dropout and parametric relu. *Journal of Computational Science*, 28:1 – 10, 2018. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2018.07.003. URL http://www.sciencedirect.com/science/article/pii/S1877750318305763.