BABD 04 Master Course – Big Data And Business Analytics

POLITECNICO DI MILANO
GRADUATE SCHOOL
OF BUSINESS

PANDA
(Prostate cANcer graDe Assessment)

An in-depth analysis of the state-of-the-art deep learning algorithms for computer vision applied to the biomedical field

Academic Tutor: Carlotta Orsenigo

Thesis of:
Gianmaria Carnazzi, Paolo Ticozzi

Academic Year 2019-2020

We decided to join a Kaggle competition about prostate cancer assessment. We want to provide a methodological approach and a guideline for image recognition tasks over an important topic which involves a very widespread healthcare issue.

Since tumor-grading analysis is quite crucial in both defining the gravity and the therapy for patients, the goal to be achieved is to help specialized professionals in their very complex job with a high-quality and stable instrument to be used as a support in their activities.
Given a biopsy the doctor will be able to define precisely and in short time the ISUP Grade related to it.

Our intention is to apply the state-of-the-art deep-learning algorithms for computer vision (such as ResNet and EfficientNet) to produce a high-quality baseline.

1. Overview and Prostate Cancer medical definition

1.1 Cancer definition

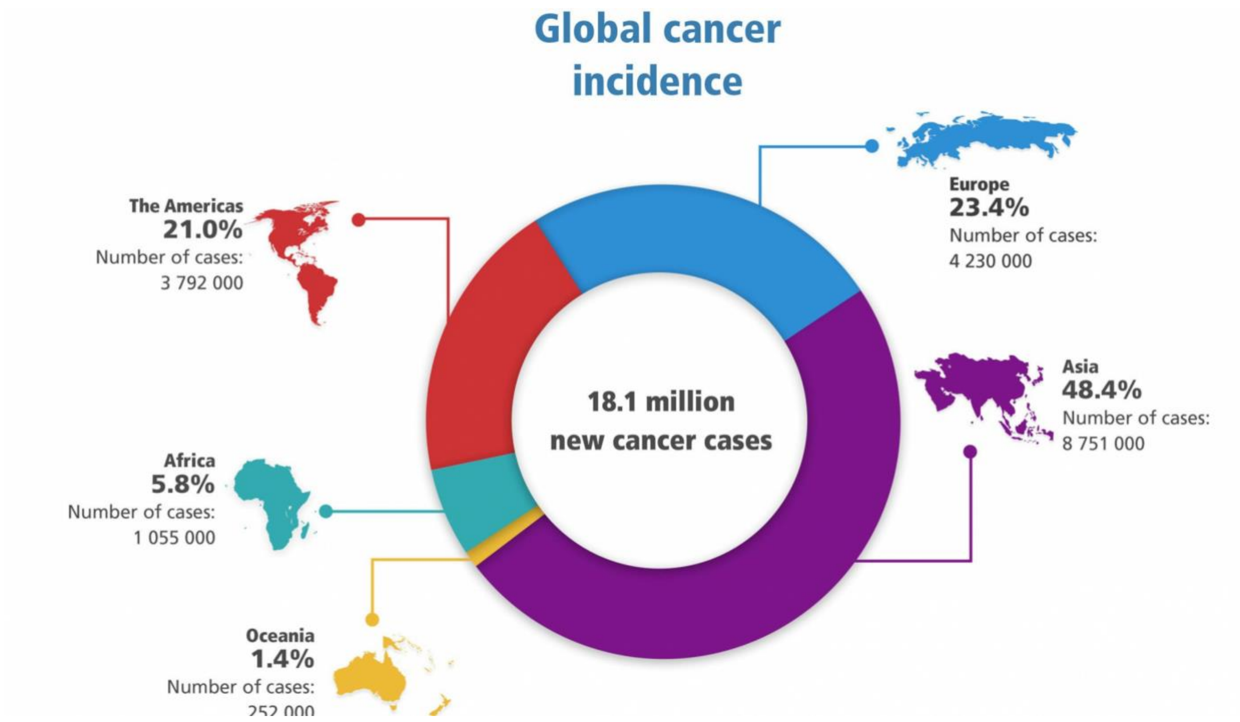
Cancer:

A term for diseases in which abnormal cells divide without control and can invade nearby tissues. Cancer cells can also spread to other parts of the body through the blood and lymph systems. There are several main types of cancer. Carcinoma is a cancer that begins in the skin or in tissues that line or cover internal organs. Sarcoma is a cancer that begins in bone, cartilage, fat, muscle, blood vessels, or other connective or supportive tissue. Leukemia is a cancer that begins in blood-forming tissue, such as the bone marrow, and causes too many abnormal blood cells to be made. Lymphoma and multiple myeloma are cancers that begin in the cells of the immune system. Central nervous system cancers are cancers that begin in the tissues of the brain and spinal cord. Also called malignancy. [1]

1.2 Statistics at a Glance; The Burden of Cancer

- The most common cancers (listed in descending order according to estimated new cases in 2018) are
  - breast cancer
  - lung and bronchus cancer
  - *prostate cancer*
  - colon and rectum cancer
  - …
- The number of new cases of cancer (cancer incidence) is 439.2 per 100,000 men and women per year (based on 2011–2015 cases).
- The number of cancer deaths (cancer mortality) is 163.5 per 100,000 men and women per year (based on 2011–2015 deaths).
- Cancer mortality is higher among men than women (196.8 per 100,000 men and 139.6 per 100,000 women). When comparing groups based on race/ethnicity and sex, cancer mortality is highest in African American men (239.9 per 100,000) and lowest in Asian/Pacific Islander women (88.3 per 100,000).
- Approximately 38.4% of men and women will be diagnosed with cancer at some point during their lifetimes (based on 2013–2015 data).

About 18 million cases around the world were estimate for 2018; of these 9.5 million were men, while 8.5 where women.
With this (quickly growing) global burden, prevention of cancer is one of the most significant health challenge of this century. It is important to define both an individual and public line to reduce and counter a very serious health threat. Indeed, the diagnosis of any form of cancer seems to be at least correlated to both individual behavior (such as smoking, high-fat diet, sedentary life) and surrounding environment (air and soil pollution, UV).



1.3 Introduction to prostate cancer

The *Prostatic Adenocarcinoma*[1] is the most common form of adult male cancer, with about 29% of the total tumoral cases verified in 2012 in the US, with about 9% of deaths cases registered in the same year.
It has been observed that there is a probability of 1 out of 6 to get a prostate cancer diagnosis, with a wide range of forms (from the most lethal ones to the more insignificants).
It is quite common to be observed in the over 50 population, with an incidence rate growth of 20% with respect to younger ages, and even 70% between 70 and 80 years old.

There are quite high variations in between races and nations, due to high correlations with hereditary and environmental factors.
It is unlike in Asia, while the most affected are the Afro-Americans. A study conducted also on Japanese population migrated in the US seems to highlight higher incidence with respect to Japanese who live in Japan.

The causes for prostate cancer are still under studies, since only basics and visual information can be retrieved by the incidence. Anyway, it seems that age, race, hormonal levels and environmental factors play a crucial role. The increase in incidence in immigrant populations from low-rated regions to highly-affected ones appears to be linked to the environmental influence, but still no specific factor can be defined as determinant.
It seems that a high-fat diet and red meat cooked on carbon might be relevant, while to prevent it might be useful a diet rich in lycopene (tomatoes are very rich of it) and vitamin D.

1.4 GRADING and STAGING; Gleason Score and ISUP Grade

The grading system is quite important in for the *prostatic Adenocarcinoma,* because grade is the best evaluation to assess the cancer danger.
Gleason Score is probably the most common system, which stratifies neoplasm[2] into 5 grades based on the glandular differentiation.
Grade 1 represents the more differentiated cancers, in which the neoplastic glands have a uniformed and approximately circled aspect, with well-defined nodules.

[1] Adenocarcinoma: is a type of cancerous tumor that can occur in several parts of the body. It is defined as neoplasia of epithelial tissue that has glandular origin, glandular characteristics, or both. Adenocarcinomas are part of the larger grouping of carcinomas [3]
[2] Neoplasm: new and abnormal growth of tissue in a part of the body, especially as a characteristic of cancer. [3]

On the other hand, Grade 5 have neoplastic glands are quite dispersed and highly infiltrated in the stroma[3].

In most of the cases, there are different glandular patterns; if very noticeable, the primary grade is assigned to the dominant pattern, while a secondary one is assigned to the rest. The two values are then combined to obtain the final score (e.g. primary grade = 3, secondary grade = 4, final score = 7); in uniform neoplasms the two grades are considered as equal, so that the final score is doubled.
Finally, in special cases with 3 or more well-defined patterns, the highest rated pattern in between the secondary grades must be summed up to the primary grade.
Following this line, the most unaggressive tumor will obtain a Gleason score of 2 (1+1), while the most harmful will obtain a score of 10 (5+5).
The assessment is typically classified based on differentiation level:

- From 2 to 6 it is considered to have an excellent prognosis
- (4+3) are moderately or scarcely differentiated
- (3+4) are moderately differentiated
- From 8 to 10 the prognosis is considered very aggressive and highly dangerous

Cancer staging is the process of determining the extent to which a cancer has developed by growing and spreading. Contemporary practice is to assign a number from I to IV to a cancer, with I being an isolated cancer and IV being a cancer that has spread to the limit of what the assessment measures. The stage generally takes into account the size of a tumor, whether it has invaded adjacent organs, how many regional (nearby) lymph nodes it has spread to (if any), and whether it has appeared in more distant locations (metastasized).

The most common staging scale is the so-called TNM (Tumor, Node and Metastasis) score assessment, which has been redefined into the ISUP grade (with a 1 to 5 scale) for prostate cancer. ISUP grade plays a crucial role into defining the right treatment for the patient.

[3] Stroma: is the part of a tissue or organ with a structural or connective role. It is made up of all the parts without specific functions of the organ - for example, connective tissue, blood vessels, nerves, ducts, etc. The other part, the parenchyma, consists of the cells that perform the function of the tissue or organ. [3]

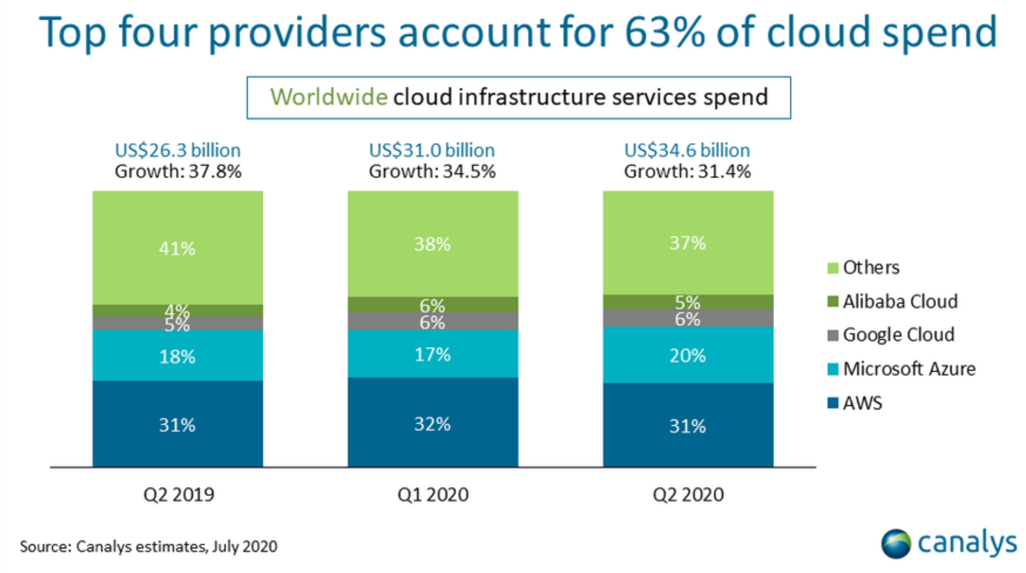
The most common therapy nowadays is the so called *prostatectomia*, which consists of a surgery therapy with total anesthesia for removing the whole prostate.
Other treatments:

- Surgery therapy
- Radiotherapy
- Hormonal therapy

2. Infrastructure

The dataset used is composed of roughly 10.000 images and 380GB, which require specific infrastructures to be properly stored and handled.
To try to run locally a model with this size input would have took too much time; nonetheless it should be available a GPU for our purposes.
The best solution is to refer to the Cloud Computing providers, which actually offer at competitive prices the opportunity to borrow highly scalable infrastructure.

2.1 Cloud computing: Gentle Introduction



Cloud computing is a large-scale and publicly accessible on a pay-per-use basis collection of compute, storage and networking resources, available through Web service calls through the Internet.
A company which provides cloud computing services have (multiple) physical datacenter(s) spread among countries to fulfill massive quantities of requests per second.
The user who needs to access the desired datacenter should be aware of data policies in the specific region to avoid highly problematic issues.

The ‘cloud computing’ term has been coined in late 2007, after the release of the very first cloud product by Amazon (EC2, which stands for Elastic Cloud Computing).
Cloud computing providers are increasing in number at very high rates, because it rapidly started to be understood the importance of this system.

IT costs are quite relevant over the balance sheet, since it requires physical infrastructure (high CAPeX costs), highly- skilled personnel to maintain it and nonetheless electricity and cooling costs. Furthermore, application deployment is a non-trivial task, which requires the configuration of hardware, OS, network, storage devices and to install al the required application software.
Cloud Computing companies offer to the customers flexible and dynamic IT infrastructures at low costs with highly configurable software services. Internet-based services are typically overestimated to deal with usage peaks, which guarantees also high stability.
It allows also to scale up and scale down the resources' usage based on the day-to-day needs.

2.2 NIST: Cloud Computing Service Models

Based on NIST (National Institute of Standards and Technologies) definition of Cloud Computing, there are three broad service categories.

- IAAS (Infrastructure-As-A-Service)
- PAAS (Platform-As-A-Service)
- SAAS (Software-As-A-Service)

2.2.1 IAAS (Infrastructure-As-A-Service)

It is a form of cloud computing which provides virtualized computing resources through the Web. It releases virtualized hardware, software, servers, storage and a number of variety of other infrastructure components on behalf of the user over the Internet.

Under the IAAS definition there are also two major network services offered by public cloud service providers.
Load balancing is the first one and provides a single point of access to multiple servers that run behind it. It distributes a set of tasks over a set of resources, to increase efficiency.
The second one is the so-called DNS (Domain Name System) service, which provides a naming system for any device that uses IP addressing networking. This procedure makes assign to each single node on the network a name, easily readable by human brains, with respect to IP addresses.

Examples: AWS (Amazon Web Services), Microsoft Azure, Google Compute Engine (GCE)

2.2.2 PAAS (Platform-As-A-Service)

It is a model to deliver apps over the Internet. A PAAS provider hosts the designed apps in its own infrastructure, both hardware and software. The main advantages for this kind of services is the high-scalability and auto-provisioning from a hand. On the other side it guarantees security and redundancy

Examples: Google App Engine, Microsoft Azure, IBM Cloud

2.2.3 SAAS (Software-A-s-A-Service)

In this model the providers releases over the Web a software in which apps are hosted by a vendor or service provider and then made available to the customers. The main advantages rely in automatic updates and patch management, easier compatibility management and administration, global accessibility. [4]

Examples: Google Docs, Salesforce.com

2.3 NIST: Cloud Computing Deployment Models

In general, cloud computing is a model in which resources are provided as general utilities that can be leased and released as general utilities, leased and released in an on-demand fashion. There are 4 types of cloud.

2.3.1 Public Clouds

It is a large- scale infrastructure available on rental basis with a pay-as-you-go base. In this case OS virtualization provides CPU isolation, network isolation and locally specific abstraction. All the services and resources are available via internet.

The public cloud deployment model is the first choice for businesses that operate within the industries with low privacy concerns

Examples: Amazon EC2 (Elastic Cloud Computing), Microsoft Azure, Google App Engine, IBM Cloud

2.3.2 Private, Community and Hybrid Clouds

When an organization sets up a virtualization environment on its own servers and fully manages it we talk about Private Clouds; in this case the company gains in security, having total control over the whole infrastructure, but required high IT investments. It is not so flexible as the public cloud could be

There is a slightly different variant of the Private one, which is the Communtiy Cloud.
It requires a group of organization to manage a single cloud. It happens when there is a partnership or a concrete and constant need of sharing info, such as in an insurance and bank case. It allows economy of scale, but there is total transparency into the federation

The last solution is represented by a mix of Public, Community and Private Clouds, to exploit all the benefits of all the deployment models and reduce the disadvantages. [5]

2.4 Google Cloud Computing (GCP)

GCP is one of the most known Cloud Computing providers with several nodes spread mainly in NA, EU and Asia (China). It is relatively easy-to-use and has a welcome free-kit of 300 $ to be used in the first year of registration of a new account, which helps the client to understand how to fluently move into the platform and the utilities he/she can navigate on and use.

The first step once the registration is terminated is to create a new project and assign the roles if multiple people can have access to the project. Assigning different roles means also restricting accessibility to the resources used in the project, since it might be that that specific role has only reading permissions or cannot open new instances.
The owner is the one who can actually modify almost anything on the project.

In our project we needed 1T of free space and 1 GPU located in the Europe-West-B node. Given the settings of the free trial (Google Cloud Free Tier) it is possible to reach up to 500GB on any physical disk (either SSD or HDD) and no GPU is available worldwide.
It is possible to obtain more resources once the upgrade of the account is done, with then the possibility of changing the so-called Quotas assigned in the nodes. Changing the Quotas might take from a couple of minutes to a week to be process.

There are two GPU Quotas that needs to be adjusted, **GPU (all regions)** to let project to have access anywhere to a GPU; **GPU (specific- region)** to have access to the GPUs in a given node. The Quotas for the specific region cannot exceed the global settings set with the first Quota.

Before creating a VM instance is good practice to activate firewall rules to let the instances to access through specific ports the requested applications. In our specific case we needed to access the GUI for Jupyter Notebook, once it has been installed on the VM itself.

Under the Firewall Rules → Create Firewall Rule it is possible to define the control of incoming (blocked by default) and outgoing traffic of an instance. Define a name for the rule, the priority within a network (lower value assigned represents higher priority), IP ranges (to set any IP with no specific ranges select 0.0.0.0/0) and finally the tcp-ip ports (e.g. 8888).

2.4.1 Set up a VM (Virtual Machine) and Create an Instance

GCP offers a service called Google Cloud Engine, which allows users to create and customize Virtual Machine based on a need-basis. The project run all in the Europe-West-B node, given the availability of specific GPUs.
The infrastructure requires at least:

- 400GB of SSD persistent disk to store unzipped data and a secondary disk of 250GB to store the zipped files downloaded from Kaggle. Alternatively, 650GB+ of SSD persistent disk
- Machine Type: n1-standard-8 (8 vCPUs, 30GB of memory)
- 1 GPU NVIDIA Tesla T4 (with 16GB of GPU memory)

Everything has been run on Debian OS, predefined by Google for deep learning tasks. The boot disk version used is a Deep Learning Image with PyTorch 1.4.0 and fastai m55 already preinstalled.
When creating an instance, it might be relevant to define also permissions for internet traffic (whether to allow incoming HTTP/HTTPS traffic or not).

It is important also to reserve a static IP address, to prevent network security problems or IP address conflict. If this task is run, GCE allows to attach a specific VM to the IP address that the VM is currently using and reserve it from that moment on. [6]

2.4.2 Set up the Instance

Once the Virtual Machine has been configured the following steps determines the configuration for Python and Jupyter Notebook.

1. Check whether any update must be run before the installation of pip package
    a. sudo apt- get update

    b. sudo apt- get –assume-yes upgrade
2. To easily manage repositories and distribution software, install software-properties-common package (readable with Debian)
    a. sudo apt-get --assume-yes install software-properties-common
3. Install python-setuptools and pip
    a. sudo apt-get install python-setuptools python-dev build-essential
    b. sudo easy_install pip

4. Install Jupyter notebook and create the config.py file to allow access through the internet
    a. jupyter notebook --generate-config
    b. sudo nano ~/.jupyter_notebook_config.py

```
c = get_config()
c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False
c.NotebookApp.port = 8888
```

5. instead of inserting access token each time, create a password
    a. jupyter notebook password
6. the access should be then available on the external IP address, at port 8888 [7]

Once Python and Jupyter Notebook is up and running, the next step is to download data from Kaggle. This is a very user-friendly task.
It requires to install the Kaggle package for Python and to download the Kaggle API associated to the user profile on the website, as a JSON file.
It is good practice to store the API in a dot file (~/.Kaggle/) in the VM, since it is a configuration file.

Once everything is ready, run the following command to run the download of the zipped data from Kaggle to the correct directory

```
kaggle competitions download -c prostate-cancer-grade-assessment -p ~/my-directory/
```

The last step is to unzip the data in the desired directory, which is a quite easy task on Linux Shell Scripting, thanks to the easy command *unzip*

The whole process for downloading and unzip this quite big volume of data takes about 3 hours with a fiber connection and an average of 75MB/s download speed-rate. [8]

Now we turn into the empirical analysis, in which we will show a pipeline for analyzing this kind of data and facing this kind of multiclassification problems. The challenge in this competition is to classify the severity of prostate cancer from microscopy scans of prostate biopsy samples. Our goal is to build a model able to classify the different form of cancer based on the ISUP grade, following the challenge rules. We will start from an explanatory analysis and then we will turn to the modeling part. We will show how to adapt some of the most used neural networks architectures for this kind of task.

3. Exploratory Data Analysis

3.1 The Dataset

The data are provided by two different sources: the Radboud University and the Karolinska Institute. They consist of four different folders: *train.csv, test.csv, submission.csv, train_images.tiff and train_label_masks.tiff*.

Files content:

[train/test].csv

- image_id: ID code for the image.
- data_provider: The name of the institution that provided the data. Both the Karolinska Institute and Radboud University Medical Center contributed data. They used different scanners with slightly different maximum microscope resolutions and worked with different pathologists for labeling their images.
- isup_grade: Train only. The target variable. The severity of the cancer on a 0-5 scale.
- gleason_score: Train only. An alternate cancer severity rating system with more levels than the ISUP scale.

[train/test]_images: Large multi-level tiff files, there are roughly 1,000 images in the hidden test set. Note that slightly different procedures were in place for the images used in the test set than the training set.

train_label_masks: Segmentation masks showing which parts of the image led to the ISUP grade. Not all training images have label masks, and there may be false positives or false negatives in the label masks for a variety of reasons. These masks are provided to assist with the development of strategies for selecting the most useful subsamples of the images. The mask values depend on the data provider.

sample_submission.csv: A valid submission file.

3.2 Cleaning

We start by looking at some numbers of the training set: we have 10616 unique ids (images), 2 data providers as we already pointed out, 6 ISUP grades and 11 Gleason scores. As previously explained, the Gleason score have almost the double of the ISUP grade values because it is built by two different numbers between 0-5 and depending on the order there is a different severity of the cancer in the sample considered.

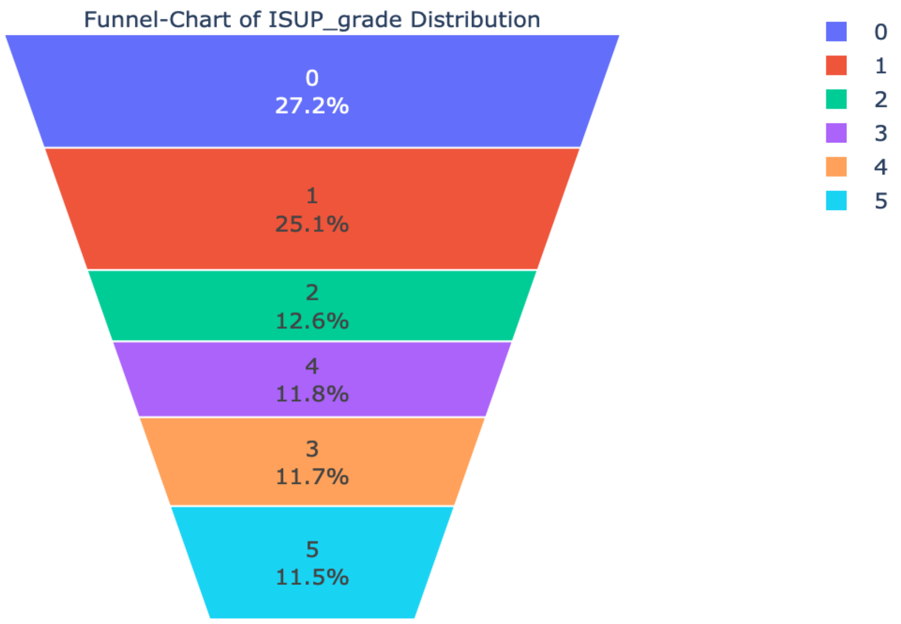
The first peculiarity to notice is that data providers use two different ways of classifying the lowest Gleason score, which basically means no cancer, that are 0+0 and “Negative”. This is the first thing to fix, the first cleaning task. We decided to adopt the 0+0 notation and so to convert the “Negative” labeled samples in that way.

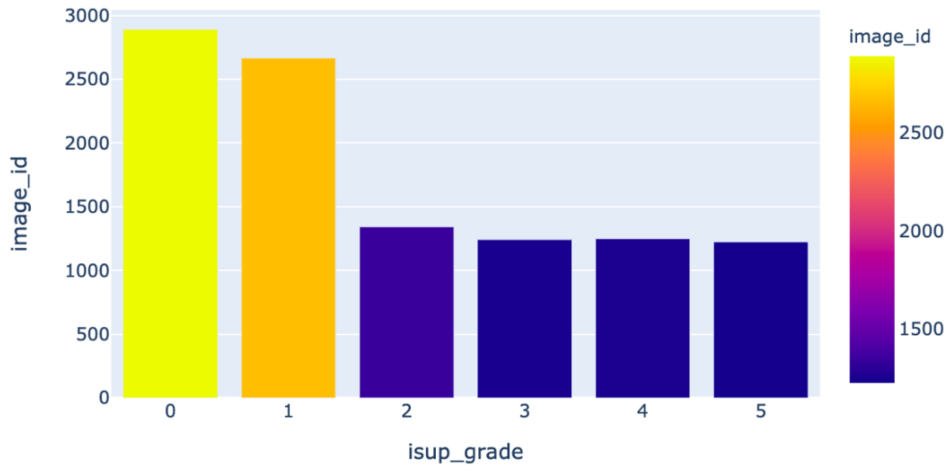
The second point regards a misleading match for a sample that points to the wrong ISUP grade, given its Gleason score, therefore we remove it.

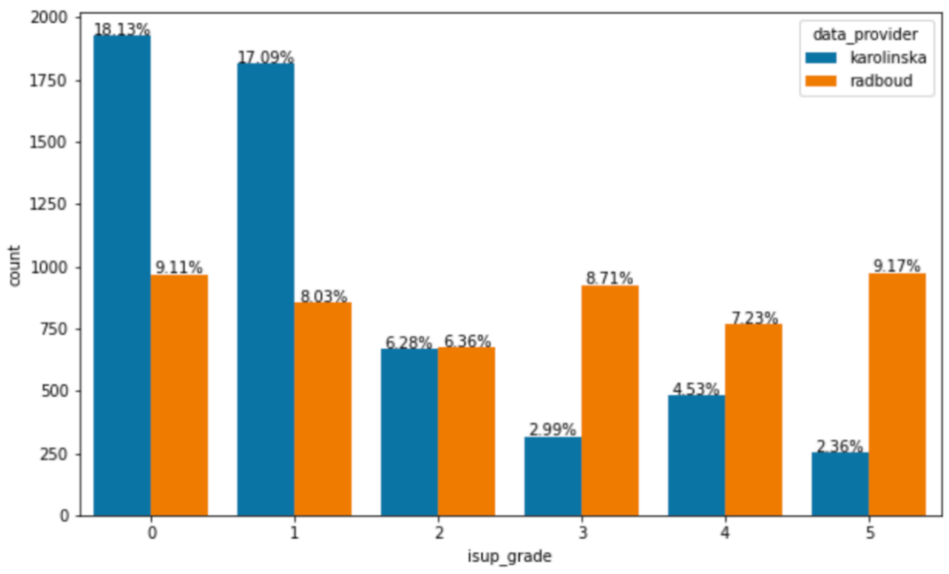
There is not much to say about the test set given that contains just few images and it can be accessed just when you are submitting. For the validation phase we will rely on the training set only.

3.3 Data Visualization

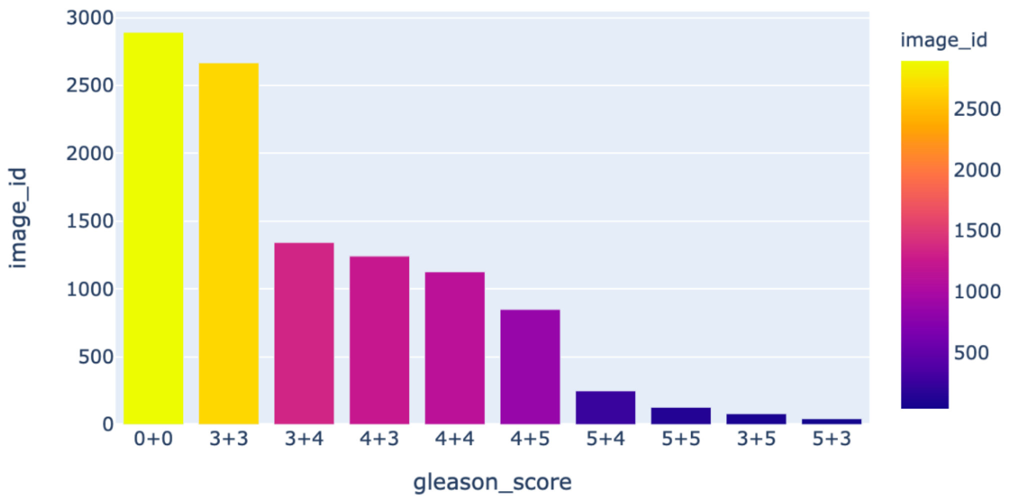
We start from one of the two possible target variables: the ISUP grade. Here the distributions of its values and other visualizations:

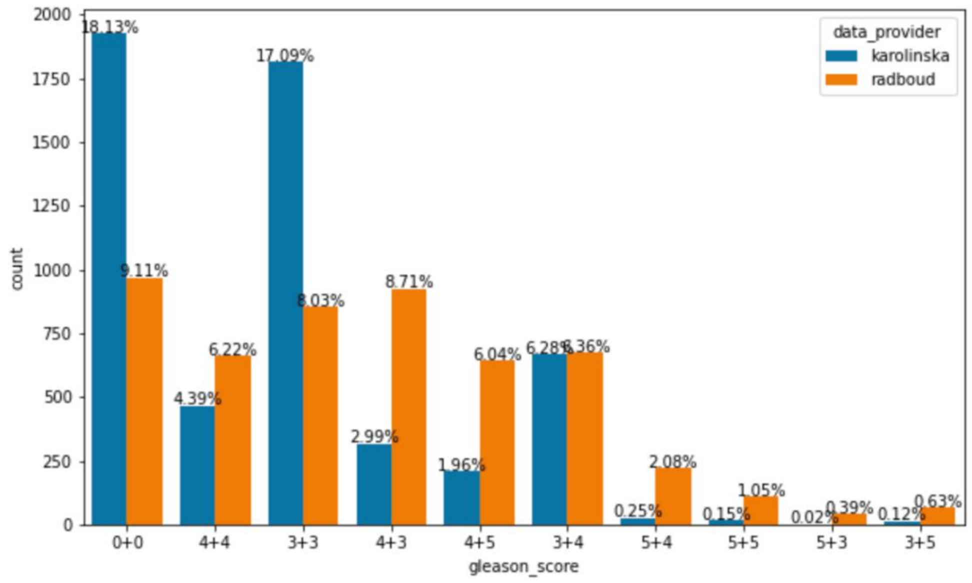






As expected, the lower the grade, the higher the percentage. The ISUP grades 0 and 1 i.e no cancer, have the highest number of values and the target class, concerning cancerous samples, is underrepresented. This is one of the biggest issue when performing machine learning tasks on Medical Data. The distribution across the two different data providers shows a higher percentage of cancerous samples for the Radboud University and a higher percentage of non-cancerous cases for the Karolinska Institute. Given the different labeling equipes of the two providers, it is not to exclude differentiating between the two providers during the training phase. Now we look at the same visualizations for the Gleason score:



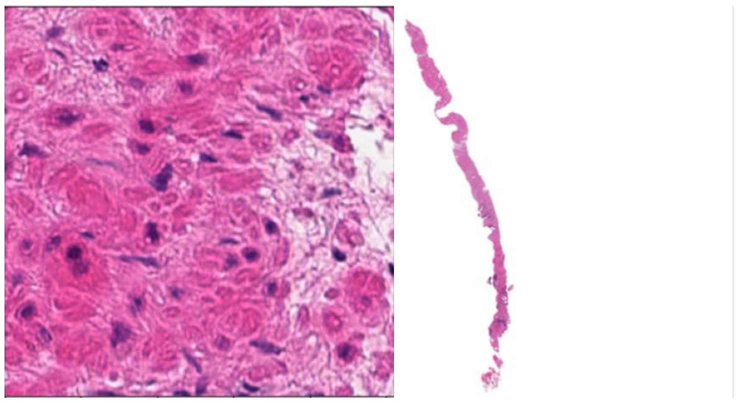


More or less the same considerations made for the ISUP grade apply here: higher percentage for lower scores and the Radboud University offers a higher percentage of cancerous samples and a lower percentage of non-cancerous samples.

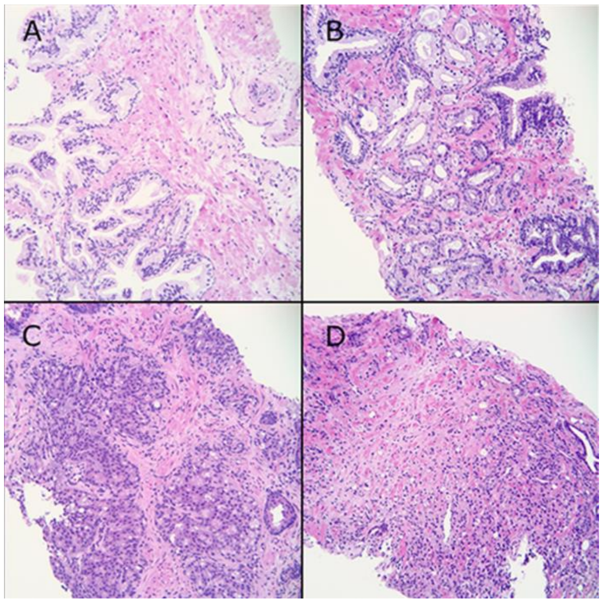
3.4 Images

The images are provided with the Tagged Image File Format (TIFF), which is a variable-resolution image format, and it is very common for loading gray-scale or colored images into page layout applications.

It is important to consider up-sampling and down-sampling images in the visualization phase, and this is what we have done and show in the following lines.



The images above represent a zoomed version and the original version of an image. Zooming can be useful to get an idea of how a cancerous tissue looks like, and to understand how to discriminate between more severe and less severe forms of cancer. Here follow some images that can help in understanding this difference.

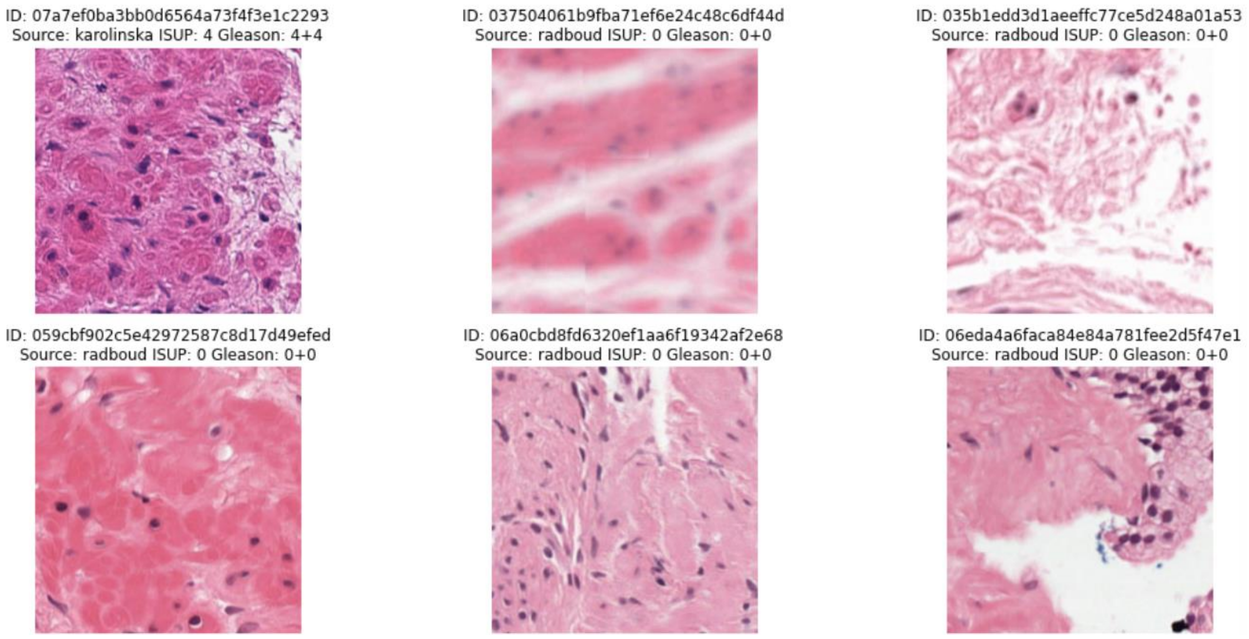


[image from Kaggle]

- **[A] Benign prostate glands with folded epithelium:** The cytoplasm is pale and the nuclei small and regular. The glands are grouped together.
- **[B] Prostatic adenocarcinoma**: Gleason Pattern 3 has no loss of glandular differentiation. Small glands infiltrate between benign glands. The cytoplasm is often dark and the nuclei enlarged with dark chromatin and some prominent nucleoli. Each epithelial unit is separate and has a lumen.
- **[C] Prostatic adenocarcinoma**: Gleason Pattern 4 has partial loss of glandular differentiation. There is an attempt to form lumina but the tumor fails to form complete, well-developed glands. This microphotograph shows irregular cribriform cancer, i.e. epithelial sheets with multiple lumina. There are also some poorly formed small glands and some fused glands. All of these are included in Gleason Pattern 4.
- **[D] Prostatic adenocarcinoma**: Gleason Pattern 5 has an almost complete loss of glandular differentiation. Dispersed single cancer cells are seen in the stroma. Gleason

Pattern 5 may also contain solid sheets or strands of cancer cells. All microphotographs show hematoxylin and eosin stains at 20x lens magnification.

Other images with labels (zoomed):


ID: 07a7ef0ba3bb0d6564a73f4f3e1c2293
Source: karolinska ISUP: 4 Gleason: 4+4
ID: 037504061b9fba71ef6e24c48c6df44d
Source: radboud ISUP: 0 Gleason: 0+0
ID: 035b1edd3d1aeeffc77ce5d248a01a53
Source: radboud ISUP: 0 Gleason: 0+0
ID: 059cbf902c5e42972587c8d17d49efed
Source: radboud ISUP: 0 Gleason: 0+0
ID: 06a0cbd8fd6320ef1aa6f19342af2e68
Source: radboud ISUP: 0 Gleason: 0+0
ID: 06eda4a6faca84e84a781fee2d5f47e1
Source: radboud ISUP: 0 Gleason: 0+0

The image dimensions are large: between 50000 and 40000 pixels and each “slide” has 3 levels whose dimensions differ based on the original image ones. Furthermore, there can be differences between biopsies with respect to the rotation and colors which can be due to different laboratory procedures.

3.5 Masks

Almost all our images have an associated mask with additional label information. The masks’ labeling directly indicate which parts of the tissue are healthy and which parts are cancerous. They can be much useful to directly classify cancerous regions of the tissue, a step forward with respect to the initial classification based on the ISUP grade or the Gleason score; here we talk about a pixel-wise classification. They can be also useful for subsampling the images.

The label information is stored in the red channel, the others are set to zero. They are not image data but simple matrices, and instead of containing values in the range of 0-255 they have a much smaller range, bounded by the number of classes identified by the data provider. indeed, depending on the data provider, the labeling is setup in different ways:

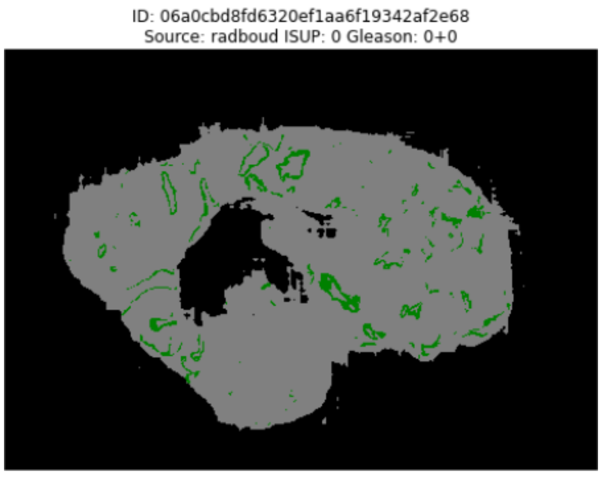
- Radboud University (prostate glands individually labeled):

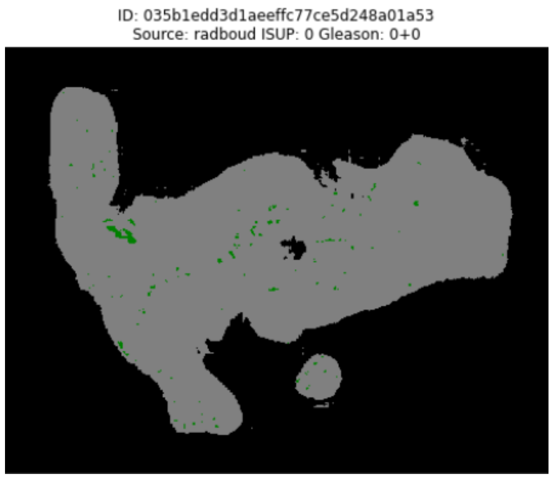
  0: background (non-tissue) or unknown
  1: stroma (connective tissue, non-epithelium tissue)
  2: healthy (benign) epithelium
  3: cancerous epithelium (Gleason 3)
  4: cancerous epithelium (Gleason 4)
  5: cancerous epithelium (Gleason 5)

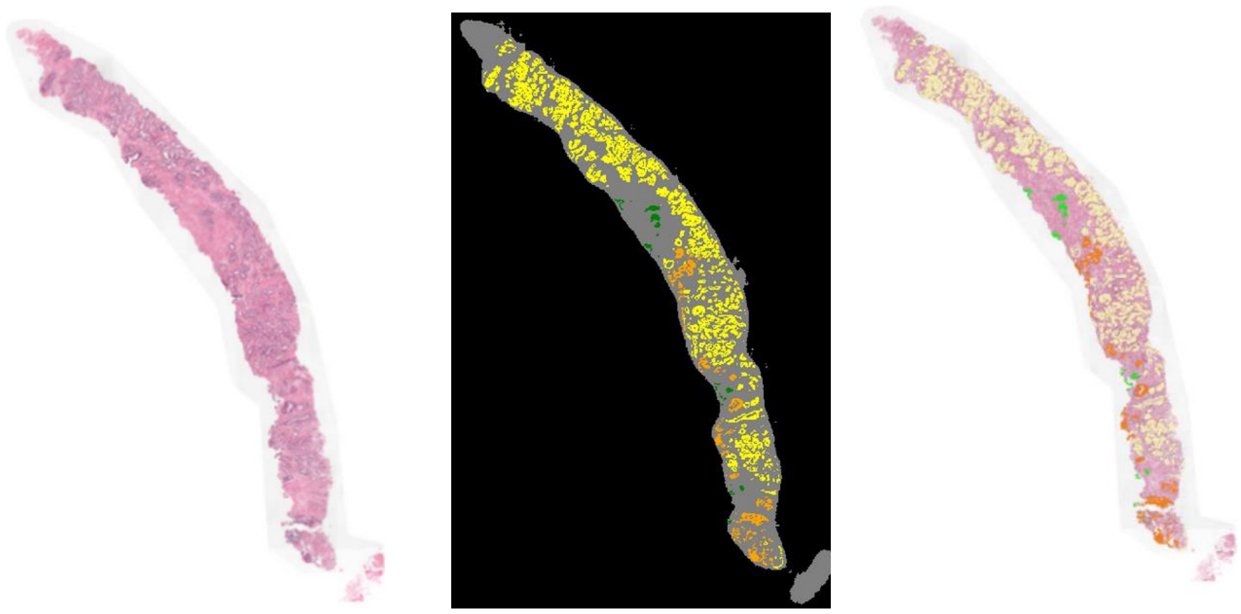
- Karolinska Institute (regions labeled):

  1: background (non tissue) or unknown
  2: benign tissue (stroma and epithelium combined)
  3: cancerous tissue (stroma and epithelium combined)

Given the small range used to classify each pixel, in order to show the image and the respective labeled regions or glands, we assigned a distinct color to each label. Otherwise, the results would show a black slide, given that the values are close to zero. Examples:


ID: 06a0cbd8fd6320ef1aa6f19342af2e68
Source: radboud ISUP: 0 Gleason: 0+0


ID: 035b1edd3d1aeeffc77ce5d248a01a53
Source: radboud ISUP: 0 Gleason: 0+0



The sequence above shows how the use of masks combined with the original images can help in directly segmenting cancerous regions and glands. In this case we have a sample provided by the Radboud University with ISUP grade of 2 and a Gleason score of 3+4. It is not a severe form of cancer but anyway it shows many infected glands, the cancer spreads all over the tissue here. We have different colors, the green and the yellow glands are the healthiest, whereas the orange/red ones are the most cancerous. This kind of composed sample could be used for building an object detection model which detects cancer in small areas of the biopsy. Nevertheless, this task is beyond the scope of this paper.

4. Modeling

This challenge falls within the image recognition field, which has witnessed a significant development in recent years with ongoing progresses in performance relative to the benchmarked tasks. Almost every year new architectures are designed in order to beat the previous best ones. The evaluation metrics rely on the performance in terms of classification scores but also on the efficiency in terms of computational costs.

For this challenge, given also that we did not have any previous experience in this field, we decided to rely on some of these architectures, slightly modified. We will show the results of ResNet and EfficientNet tested on the data provided. Using pre-defined architectures opens the door of the so-called transfer learning field, where those architectures are adapted to new kind of data. There exist different ways of adapting them: fine-tuning, feature extraction and mixed techniques like “Learning without forgetting” or “joint training”.

4.1 Transfer learning

Few people train a Convolutional Network from scratch, because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset, like ImageNet, and then use it either as an initialization or a fixed feature extractor for the task of interest. There are three ways of using such pretrained ConvNets:

- **Fine-tuning**: this approach consists in replacing and retraining the classifier on top of the ConvNet and also fine-tuning the weights of the pretrained network by continuing the backpropagation. It is possible to do it for all the layers or to keep some of the earlier layers fixed and only fine-tune some of the others. This second option is much more adopted because earlier layers of a net usually contain more generic features, which means that they hold a generalization power, they can be useful to many tasks, whereas later layers becomes progressively more specific to the details of the classes contained in the original dataset.
- **Feature extraction**: It consists of using a pretrained ConvNet, removing the last fully-connected layer and then treating the rest of the ConvNet as a feature extractor for the new dataset. The extracted features are called codes. Once they are extracted, they are used to train a linear classifier with new data.

- **Pretrained models**. Given the high costs of training, it is common to see people release their final ConvNets for the benefit of others who can use the networks for fine-tuning.

How can we decide what type of transfer learning to perform? There are several factors, but the two most important ones are the size of the new dataset and its similarity to the original dataset. From this consideration we have 4 different scenarios to analyze, depending on the new dataset:

1. **Small and similar***:* since the data is small, it is not a good idea to fine-tune the ConvNet due to overfitting concerns. Since the data is similar to the original data, we expect higher-level features in the ConvNet to be relevant to this dataset as well. Hence, the best idea might be to train a linear classifier on the codes.
2. **Large and similar**: Since we have more data, we can have more confidence that we won't overfit if we were to try to fine-tune through the full network.
3. **Small and different**: Since the data is small, it is likely best to only train a linear classifier. Since the dataset is very different, it might not be best to train the classifier form the top of the network, which contains more dataset-specific features. Instead, it might work better to train the SVM classifier from activations somewhere earlier in the network.
4. **Large and different**: Since the dataset is very large, we may expect that we can afford to train a ConvNet from scratch. However, in practice it is very often still beneficial to initialize with weights from a pretrained model. In this case, we would have enough data and confidence to fine-tune through the entire network.

There are few other considerations to make when applying transfer learning. For example, using a pretrained network means being constrained in terms of the architecture that can be applied for the new dataset. It is not possible to take out Conv layers from the pretrained network. In exchange, images of different spatial size can be easily used. Another consideration regards the learning rate: It's common to use a smaller learning rate for ConvNet weights that are being fine-tuned, because we expect that the ConvNet weights are relatively good, so we don't wish to distort them too quickly and too much.

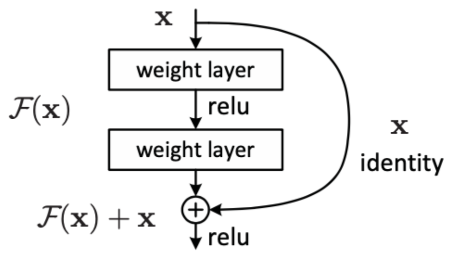
4.2 ResNet

We decided to implement this network architecture because of the results already showed in other tasks. Resnet presents a residual learning framework mainly built to ease the training. It adopts a shortcut structure which makes it fall within the class of highway networks.

As the name suggests, the idea is not to fit directly an underlying mapping but a residual mapping which is shown to be easier to optimize. The formulation can be realized with feedforward neural networks with the so-called “shortcut connections”, which are those skipping layers. In Resnet, these connections are identity connections, or mappings, and their output are added to the output of the stacked layers. It is shown that this architecture eases the optimization process and leads to gains in performance. It also addresses the problem of degradation: accuracy saturation as the net gets deeper. Thanks to the identity mappings of the shortcuts, the performance cannot be worse than the shallower version of the network. This is the building block formulation:

$$y = \mathcal{F}(x, \{w_i\}) + x$$

Here x and y are the input and output vectors of the layers considered. The function $\mathcal{F}(x, \{w_i\})$ represents the residual mapping to be learned. This function can represent different types of layers, convolutional or linear, and their number can vary from one to three or even more. This is an example of a residual block:



The generalization form of a residual block is this one:

$$y = h(x_l) + \mathcal{F}(x_l, w_l),$$

$$x_{l+1} = f(y_l)$$

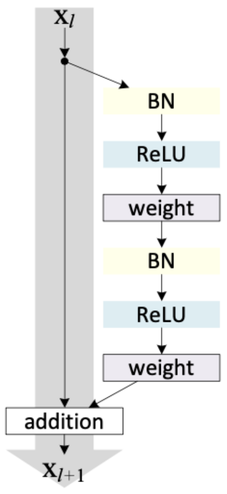
where x_l is the input feature, w_l is the set of weights and $\mathcal{F}$ is the residual function again (a stack of convolutional layers, for example), *l* is the residual unit, f is ReLU or the identity, and h is the identity. Form this formula we can recursively get to a general form for each input of a given residual unit, if f is the identity:

$$x_L = x_l + \sum_{i=l}^{L-1} \mathcal{F}(x_i, w_i)$$

Computing the gradient, we get:

$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L} \frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L} \left(1 + \frac{\partial}{\partial x_l} \sum_{i=l}^{L-1} \mathcal{F}(x_i, w_i)\right)$$

Hence, the gradient can be decomposed into two additive terms, a first term $\frac{\partial \mathcal{E}}{\partial x_L}$ that propagates information directly without concerning any weight layers, and the other term through the layers. It is important to underline that the first term is unlikely to be canceled out for a mini-batch, because the second term cannot always be equal to -1 and so the gradient does not vanish even with small weights. This is the crucial point of highways networks: a gradient propagation which tackles usual optimization problems like the vanishing of the gradient itself. The condition of f is maintained by rearranging the structure of the blocks, adopting the so-called full pre-activation, which implies that there is no activation function before or after the addition and so the backpropagation advantages explained above are held.



The main differences from a design standpoint lie on the possible settings of the shortcut/skip connections and the activation functions. The most used setting for the activation is the full pre-activation shown above and that differs from the original version which puts ReLU after the addition.

What about skip connections? There are some possible modifications and the most obvious one is introducing a scaling factor on the first additive term which propagates the gradient without touching the weight layers:

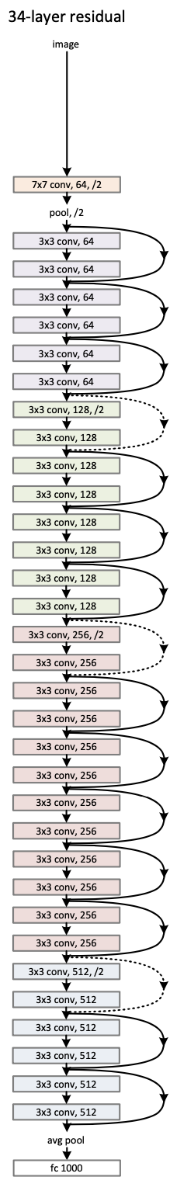
$$x_{l+1} = \lambda_l x_l + \mathcal{F}(x_l, W_l)$$

$$x_L = \left(\prod_{i=l}^{L-1} \lambda_i\right) x_l + \sum_{i=l}^{L-1} \hat{\mathcal{F}}\,(x_i, W_i)$$

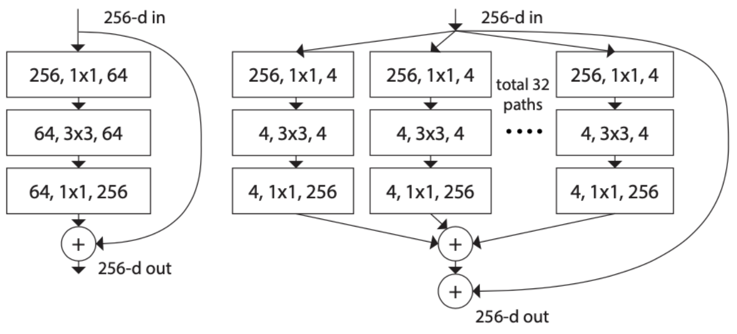
Where λ_l is our scaling factor and $\hat{\mathcal{F}}$ absorbs the scalars into the residual function.

For a large network, if $\lambda_i > 1$ for all i, the first term (product one) can get exponentially large; if $\lambda_i < 1$ for all i, the factor can get extremely small and vanish, and this will block the gradient propagation from the shortcuts and will forces it to flow through the layers. Therefore, finding the optimal value is crucial, but is not easy. If, instead of introducing a scaling factor, we adopt more complex transformations like a 1 x 1 convolution, or implement gating strategies, things get even more complicated. The product term could also hamper the training, together with blocking the gradient flow. That's why for skip connections, the best solution is the original one: identity mapping. Residual networks are a very good solution for image classification task like ours, they can ensure a good performance in terms of accuracy and efficiency thanks to their architecture and that's why they represent a logic choice.

As other deep learning techniques, there exists many versions of this architecture, developed by different authors. They differ for the number of layers (the depth of the network) or for the introduction of different modules; there exists also a combination of the residual architecture with the inception module. As the depth increases, the accuracy increases, but also the FLOPS (floating point operations per second), so the computational costs. This is the usual tradeoff in deep learning and that's the reason why we implemented more than one version of the algorithm, with a different number of layers. Unfortunately, we do not have high resources at our disposal, so we tried the best we could. Here follows an example of the architecture with 34 layers.

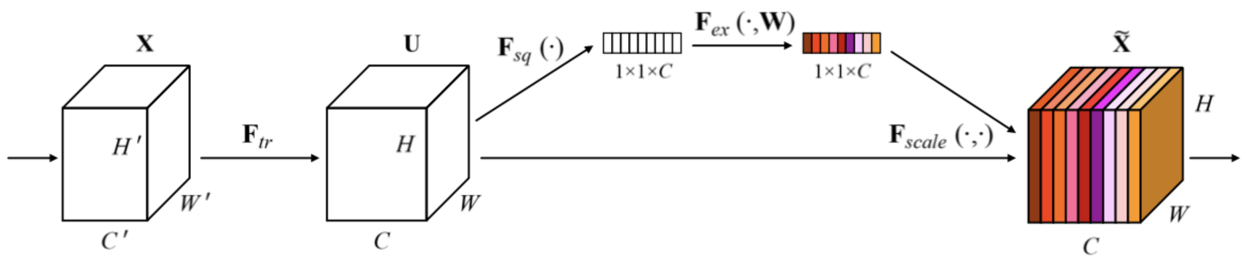


Now we briefly introduce ResNext and the Squeeze and Excitation module which we tested to improve our results. ResNext comes from an aggregation concept which aims to introduce a new way of improving performance which does not imply increasing the depth of the network or the number of hyperparameters. The objective was to increase accuracy while maintaining or even reducing the computational complexity of the model. The authors took inspirations, among others, from the *split-transform-merge* strategy of the inception networks and the grouped convolutions strategy, like in AlexNet. The main concept is *cardinality*, which is the size of the set of transformations applied.



The figure above shows the original version of a residual block and the corresponding “Next” version with cardinality equal to 32. They share more or less the same complexity, but there is an increase in accuracy.

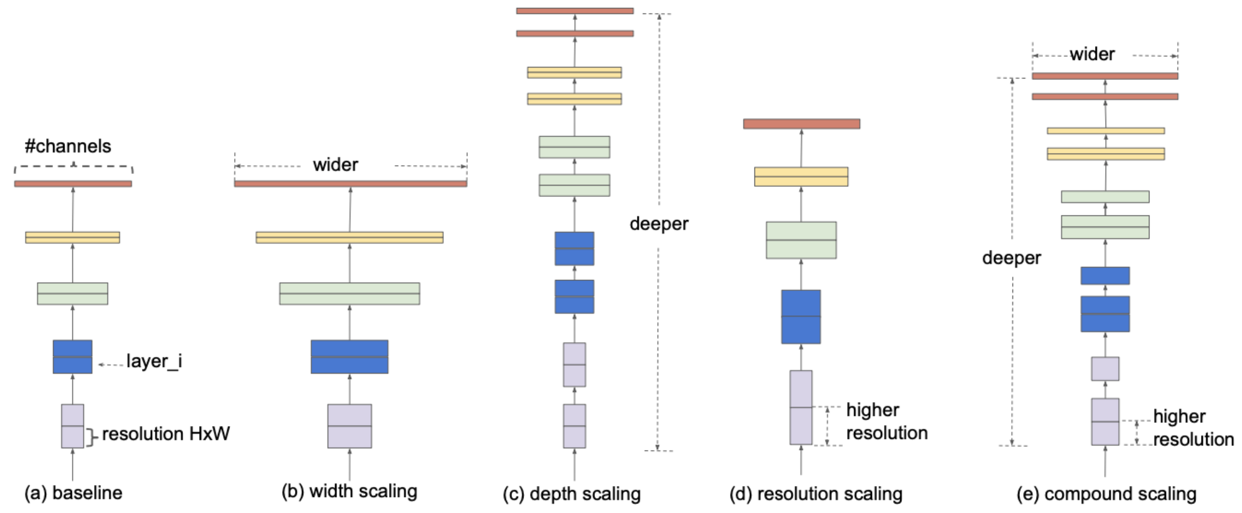
Now we turn our attention to the Squeeze and Excitation module, which we used in combination of ResNext for our task. This architectural unit (SE block) introduced a new aspect of network design: the relationship between channels. The goal was to improve the quality of the network representation by explicitly modeling the interdependencies between the channels. The idea is that channels can have meaningful relationships which carry representational power of the data. For any given transformation of the input to the feature maps we can construct a corresponding SE block to perform feature recalibration. The features are first passed through a *squeeze* operation, which produces a channel descriptor by aggregating feature maps across their spatial dimensions (H × W). The aggregation is followed by an *excitation* operation, which takes the embedding as input and produces a collection of per-channel modulation weights, which are applied to the feature maps to generate the output of the SE block. It is possible to construct an SE network (SENet) by simply stacking a collection of SE blocks, or they can be used as a drop-in replacement for the original block at a range of depths in the network architecture, as we did with ResNext.



4.3 EfficientNet

ConvNets scaling is one of the main procedures in deep learning. Usually the network is scaled up if more resources are available. EfficinetNet represents a family of networks that make balanced scaling their warhorse. They introduce a new scaling method for uniformly scaling all dimensions of depth, width and resolution using a simple compound coefficient. thanks to this method, they reach the best performance in terms of accuracy. They represent the state of the art of convolutional neural networks.

Scaling up is a widely used method to achieve better accuracy. There are many ways to do it, the most common are by depth, by width or by resolution. The crucial point is that, before EfficientNet, just one dimension at time was scaled. As we said earlier, the fortune of this new family of networks is based on a compound scaling method, they uniformly scale all dimensions with a set of fixed coefficients.



The coefficients are set taking into account the computational resources available and the input dimension. If the image is bigger, the network should have logically more layers to increase the receptive field and more channels to capture more patterns.

Now we briefly look at the problem formulation:

$$\mathcal{N} = \bigodot \mathcal{F}_i^{L_i}\left(X_{\langle H_i, W_i, C_i \rangle}\right) \quad where\ i = 1 \ldots s$$

Where $\mathcal{F}_i^{L_i}$ indicates that the layer $\mathcal{F}_i$ is repeated L_i times in stage i (a stage is a combination of layers which is repeated multiple times to build the architecture, sometimes with some modifications). Inside the function we have the input tensor $X_{\langle H_i, W_i, C_i \rangle}$ with its dimensions' values. The object of model scaling is to explore the space of the dimensions for each layer and find the optimal values, subject to certain constraints, usually resource constraints. Therefore, we talk about a maximization problem:

$$max_{d,w,r} \quad Accuracy\big(\mathcal{N}(d,w,r)\big)$$

$$s.t. \qquad \mathcal{N}(d,w,r) = \bigodot \hat{\mathcal{F}}_i^{dL_i}\big(X_{\langle r\hat{H}_i, r\hat{W}_i, w\hat{C}_i \rangle}\big)$$

$$Memory(\mathcal{N}) \le target\ memory$$

$$FLOPS(\mathcal{N}) \le target\ flops$$

Where d, w, r are the scaling coefficients. The main problem is that they depend on each other, that's why conventional network scale in just one dimension, trying to get the best improvement. Scaling just one dimension still gets some gains, but the problem is with big models: the bigger is the model, the lower the gain in accuracy. With EfficientNet and its compound scaling these coefficients are jointly modified. The intuition is logical: if we have a higher resolution, we need also more depth and width to get the right features and patterns with bigger images. The compound scaling method works using a compound coefficient ϕ to uniformly scales width, depth and resolution:

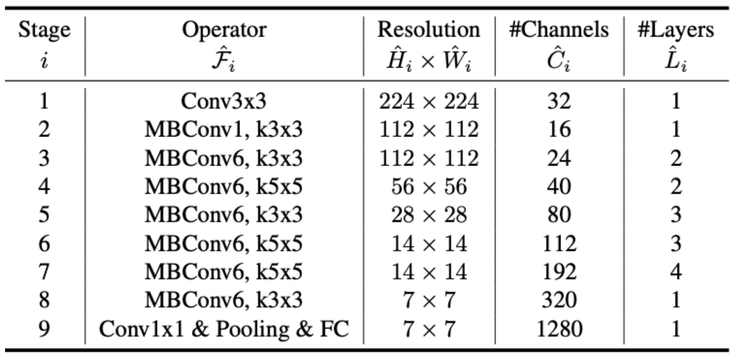
$$depth{:}\, d = \alpha^{\phi}, width{:}\, w = \beta^{\phi}, resolution{:}\, r = \gamma^{\phi}$$

$$s.t.\ \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \ge 1, \beta \ge 1, \gamma \ge 1$$

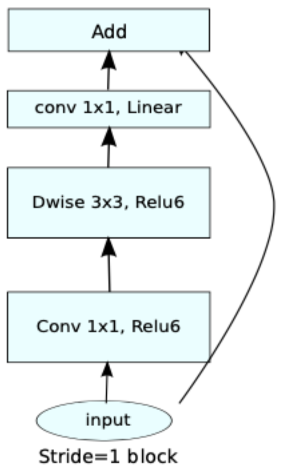
Where α, β, γ are constants determined by a grid search. ϕ controls how many more resources are available for scaling, the others specify the assignment to the different dimensions. FLOPS are proportional to d, w^2, r^2 and since convolutions usually dominate the computational costs in ConvNets, the increase of FLOPS would be something like this: $(\alpha \cdot \beta^2 \cdot \gamma^2)^{\phi}$. Therefore, the first constraint ensures that for any new ϕ, the FLOPS will approximately increase by 2^{ϕ}.

Together with the compound scaling we have a new architecture inspired by MnasNet, an architecture for Mobile, which tries to optimize both accuracy and the latency, and MobileNetV2. Here follows the baseline.

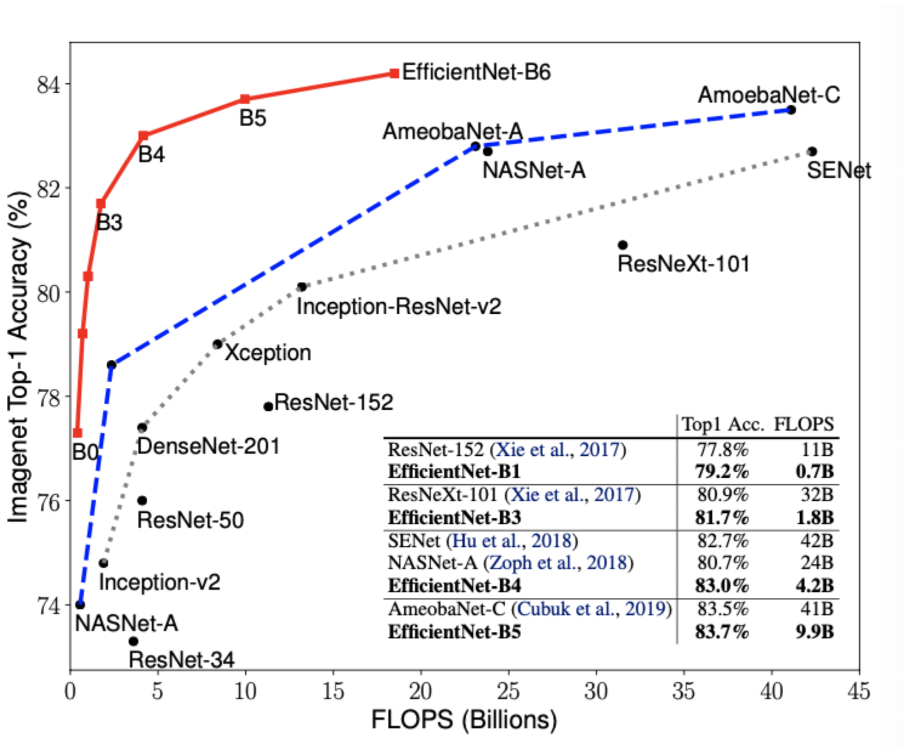


Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

The architecture is based on the inverted residual block introduced by MobileNetV2. The main difference is inspired by the intuition that bottlenecks contain all the necessary information, so shortcuts are directly built between them. Furthermore, linearity acquires a major role. The idea comes from the understanding of the risk of using non-linearities, which tend to destroy too much information. The so-called manifold of interest can actually lie in a subspace of the input space which can be captured directly by a linear layer, saving computational resources, and avoiding losing information, given that non-linear transformations can make points collapse into each other. In Mnasnet, a subsequent work of MobileNetV2, the diversity of layers gets a main role. Using reinforcement learning, the best architecture is found in order to achieve a Pareto optimal solution, which means in this case: either getting the highest accuracy without increasing latency (a more precise measure of computational costs), or the lowest latency without decreasing accuracy.



Starting from the baseline (EfficientNet-B0) the compound is applied: a small gird search finds the optimal values: $\alpha = 1.2,\ \beta = 1.1,\ \gamma = 1.15$, then the network can be scaled up with different ϕ to obtain the scaled versions of the model, from EfficientNet-B1 to EfficientNet-B7.



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.2%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.7%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**83.0%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.7%**	**9.9B**

As the graph above shows, the model outperforms the others in both accuracy and computational cost efficiency. The higher the version, the higher the scores and logically the FLOPS, given that the network is scaled up. This is the usual tradeoff to deal with.

4.4 Training ResNet

ResNet is our first choice. We tested different versions, which mainly differ for the numbers of layers, with a maximum of 101. We already talked about the advantages of this kind of architecture, we just tested deeper versions to improve our results, but it turned out that that this time, there is no significant advantage in using more layers. Actually, given that we have few classes, it is not necessary, or it is even redundant to use many layers as we did, with the risk of falling into a saturation trap, but we tried anyway to get a better performance.

Preprocessing: each sample passes through a sequence of transformations which are adopted to introduce some randomness in the data and to adjust them for the training of this particular architecture. The first two transformations fall within the segmentation process of the images, whose aim usually is to increase the number of samples by introducing transformed versions of the original data in order to train the network to different forms/shapes of inputs. In our case though, we do not increase the number of images, but we just set a probability which determines if transforming or not the input, introducing a more realistic randomness part in the sampling phase. We have a horizontal flip and a vertical one with probability of 0.5. it follows a normalization required to feed the network which adopts mean and standard deviation chosen based on the ImageNet dataset on which the model has been pretrained on. Remember that we rely on transfer learning.

The data are split into 4 folds using a Stratified k-folds strategy which splits data in a balanced way, which means that classes among different folds are equally represented. Three folds are used as training set, the last one as validation/test set. Usually this kind of strategy is applied in order to implement cross-validation, but, given the relatively small size of the dataset, we decided to use a traditional approach splitting the data in two set and just rely on the folds strategy to equally balance them.

The network: we applied fine tuning on ResNet with 18, 50 and 101 layers, and we completely reset the last fully-connected layer. We decided to introduce a ReLu hidden layer, with a dropout rate of 0.5, and to bound it by two linear layers. It follows a Softmax activation function. As last attempt, we implemented a ResNext with 50 layers and a Squeeze and Excitation module attached at each block. We did not apply it to the 101 layers version because we realized, looking at the feature maps, that the network seems not to get any pattern from many layers.

The optimization: the first optimization setting we tried was built with Adam as optimization algorithm, the original version without AMSGrad or any other modification, and as starting parameters we used:

- **betas** - coefficients used for computing running averages of gradient and its square: 0.9, 0.999
- **eps –** term added to the denominator of the exponential running average to get stability and avoiding a big jump at the beginning: 1e-8
- **weight decay:** 0
- **learning rate:** 3e-4, which is the one usually suggested when using Adam, divided by the warmup factor

The learning rate schedule was based on a reduction on plateau. This scheduler reduces the learning rate when the chose metric stops improving. Models often benefit from reducing the learning rate by a factor of 2-10 once the rate falls into a plateau. This scheduler reads a metrics quantity and if no improvement is seen for a "patience" number of epochs, the reduction is applied. We set a patience of 2 and a reduction factor of 0.5 starting from the default of 0.1, it seems to be the more efficient setting. We also tried other learning rate scheduler, for example the cosine annealing, which we explain later with EfficientNet, but we do not get any improvement. Lastly, as loss function we used a simple Cross-Entropy and a quadratic weighted kappa as evaluation metric (1), following the challenge rules. Unfortunately, we did not get good results. We hit the 60% with 101 layers, but looking at the features maps of the layers we noticed that the majority, did not get any patterns, they are basically useless. Turning to the 50 layers architecture we get a worse performance, but with a meaningful architecture.

Quadratic weighted kappa

$$k = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}} \qquad (1)$$

Where k is the number of classes, $O_{i,j}$ is matrix that collects the misclassifications of each class by the classifier, $E_{i,j}$ is the expected outcomes matrix, computed with the outer product between the actual outcome and the predicted one, and $w_{i,j}$ is the weight matrix, which, in the quadratic case, is computed as follows:

$$w_{i,j} = \frac{(i-j)^2}{(N-1)^2}$$

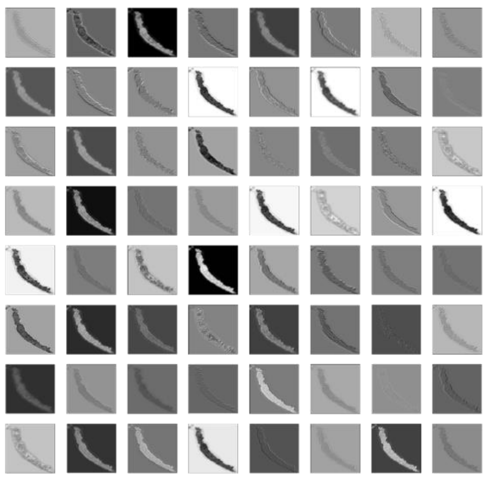
The weight matrix is pretty useful in multi-classification problems like ours, because it gives more weight to misclassifications that happens for categories very different from each other. The higher the difference, the higher the weight, because not all misclassifications are equal.

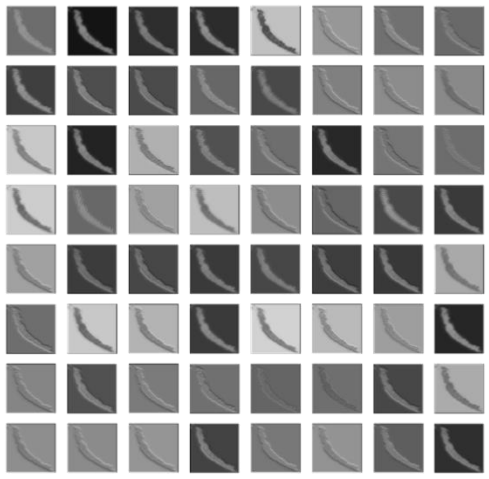
Basically, the numerator measures a weighted sum of misclassifications, whereas the denominator measures the expectation of casual matches or mismatches between the actual and the predicted outcomes. The underlying idea is to take into account randomness, which can play a role in the final results, in favor or not of the classifier.

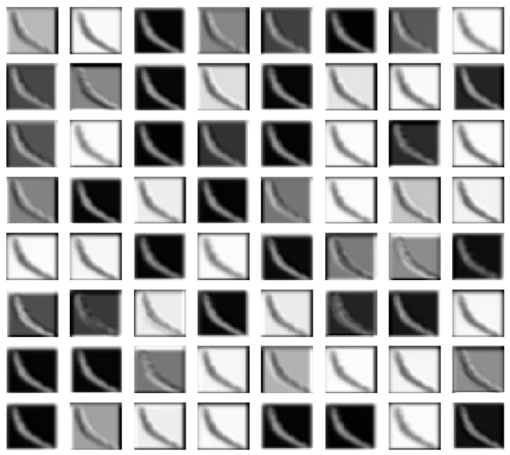
Conclusion: we tested four different versions of ResNet: 18 layers, 50 layers, 101 layers and a ResNext of 50 layers with a Squeeze and Excitation module (SE_ResNext50_32x4d). We played a little bit with the batch size until we decided to set it at 2. The Squeeze and Excitation module ResNext is the most performing one (score: 62%), but it has been overcome by our champion of the game: EfficientNet.

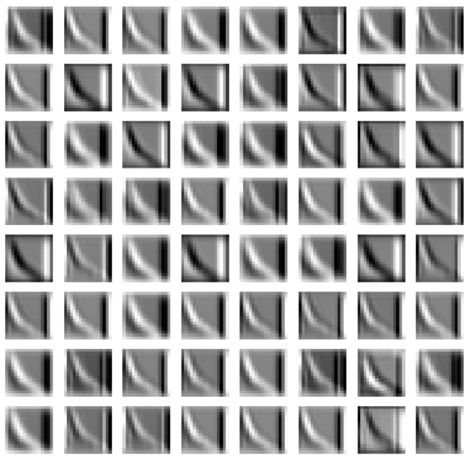
4.4.1 Visualizing and understanding the network

In this section we show some famous techniques used to understand how the network works and visualize its transformations and processes in general. The first thing to do when going inside the network is visualizing the weights (kernels) of the convolutional layers. Actually, they are not that meaningful for visualization, but they are key players of the game. The second step, and the most important one, is looking at how they transform inputs, so looking at the feature maps. Here we show an example using our ResNet50 model, starting from the layer 0 to the last one.









As we go deeper, the feature maps get less clear, because the transformations keep accumulating. As we already said, with ResNet101, after a certain number of layers, the network seemed not to get any pattern from the data, that's why we also decided to use a smaller architecture. Probably the layers could be increased up to the 60th layer to improve the results but given the performance of EfficientNet we did not get that far in transforming the architecture. Now we look at another meaningful visualization: saliency maps. The idea behind saliency maps is understanding how each pixel of an input image affects the final score of the network. So, given an image I_0, a class c, and a classification ConvNet with the class score function $S_c(I)$, we want to rank the pixels of I_0 based on their influence on the score $S_c(I_0)$. Let's consider the simple example of a linear classifier:

$$S_c(I) = w^{\mathrm{T}} I + b$$

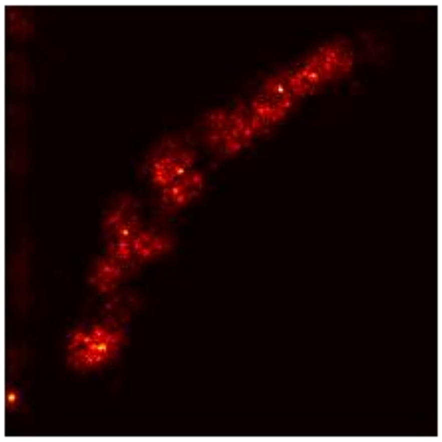
In this simple case, it is easy to see that the weights directly represent the influence of each pixel on the final score. Nevertheless, in convolutional neural network the model is highly non-linear and the equation above does not hold. However, we can still rely on the Taylor expansion to approximate the function in a neighborhood of the input:

$$S_c(I) \approx w^{\mathrm{T}} I + b$$

Where *w* is the derivative of the input image at a certain point:

$$w = \frac{\partial S_c}{\partial I}\Big| \quad wrt\ I_0$$

The higher the derivative, the higher the influence, which means that that pixel would be changes less with respect others with a lower value, because it is meaningful by itself for the classification. Example:



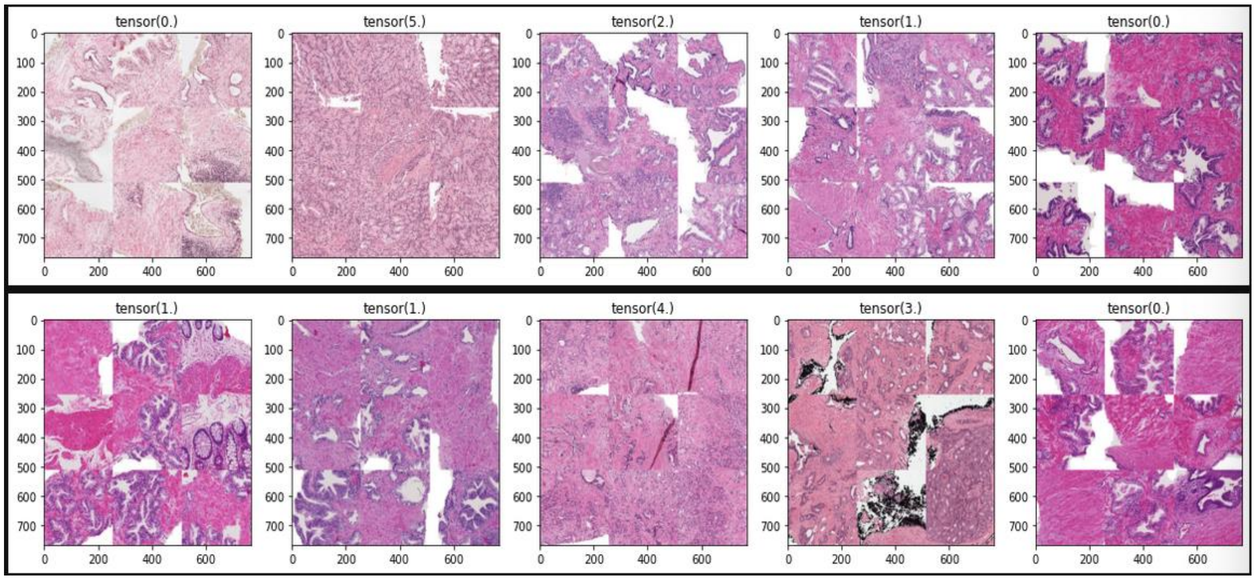
In our case this technique is less powerful because the object is easily recognizable and usually it is more difficult to differentiate tissue regions just by looking at this kind of output. Anyway, it is useful to see if the network goes in the right direction and here, actually, it can be seen that certain tissue regions are more important than others.

4.5 Training EffiicientNet

Now we move to the second algorithm tested. We already talked about the theoretical advantages of EficientNet, it is time to show that they apply in practice. We implemented the first version B0 because of the computational costs implied by the training, we do not have a high budget at our disposal. As the graph above shows, it is probable that the accuracy can improve a lot by applying other versions. Nevertheless, we obtained quite good results, better than ResNet, and that's also why we stopped to this version.

Preprocessing: again, we split the data using Stratified k-fold strategy, which enables us to have a certain number of folds with almost the same class percentages. As transformations composition before feeding the network we used a transposition, a vertical and a horizontal flip with default probability of 0.5. With Resnet we used some transformations to adapt the images to the network, whereas here we just add this composition to introduce some randomness in the data to make our model more robust to different image formats. Batch normalization is already part of the network modules, so we did not apply any kind of normalization or standardization before.

Tiling: the main difference with respect to the previous preprocessing strategy lie on the tiling process. This is the real novelty of this application. Given the relatively large size of the images and the fact that the majority of the space is blank, we optimized by tiling the images. In practice, we set a priori a number, let's say N, which represent the number of tiles we want from each image and the tile size; logically, the higher the number N, the more accurate the representation, but also the risk of getting blank and useless tiles. Using the image dimensions and the tile size decided, some padding factors are computed in order to pad the image, if necessary, to fit the tile size desired. After some reshaping and transformations, you end up with a certain number of tiles that fit the tile size set a priori, it does not correspond to the number desired, because the idea is that you want also to select the tiles, to cut the blank ones. So, you take just the N tiles with the smallest sum. The reasoning behind is that: the smallest pixel value sum, the darker the color and so the higher the probability of selecting tissue regions, which are the ones we are interested in. The drawback of this strategy lies, as usual, in the computational cost. Indeed, each tile feeds independently the network and then it is concatenated with the others to get the result for the image. Therefore, the running time tends to increase. Alternatively, the tiled image can be directly used as input. As mentioned before, we could also try to use the masks for subsampling the images, but given the differences between the labeling procedures of the two data providers, and the possible increase in human mistakes, we decided to automatize the subsampling starting from the biopsy. Here follow some samples with 12 tiles:



Training: the procedure follows the same steps of ResNet: the final fully-connected layer is reset in order to train it with new data. Therefore, we used the pre-trained model and we fine-tune it for getting predictions for our dataset. The quadratic weighted kappa is used as evaluation metric, following the challenge rules, and a sigmoid is applied in the output layer. We used Adam as optimization algorithm given that it is basically the state of the art with its different versions, and considering the limited resources we did not try other methods, but we tested the AMSGrad version which do not rely on the exponential running averages of past gradients and it applies a sort of long-term memory on the past calculations to converge to the global optimum. It is supposed to perform better, but we did not notice a significant difference. Anyhow, as starting parameters of Adam we used a similar setting to ResNet.

The learning rate annealing is one of the most important things of the optimization phase of the network. We used a cosine annealing procedure, which means that we used a cosine function as annealing schedule for our learning rate. The underlying ideas is that there exist periods with high learning rates and periods with low ones. The function of the periods of high learning rates in the scheduler is to prevent the learner from getting stuck into local minima or saddle points of the objective landscape, whereas the function of the periods of low learning rates should allow to converge to a near-true-optimal point. The periodicity of the function is the key.

$$\eta_t = \eta_{min}^i + \frac{1}{2}\left(\eta_{max}^i - \eta_{min}^i\right)(1 + \cos(\frac{\mathrm{T}_{cur}}{\mathrm{T}_i}\pi))$$

Where η_{min}^i and η_{max}^i are ranges for the learning rate, the former is usually equal to 0, and T_{cur} accounts for how many epochs have already been performed since the last restart. We kept the min-max range fixed for simplicity, but it could be decreased at every restart, or changed in any other optimal way. T_i can be multiplied by a certain factor or kept fixed to make the learning rate decrease over time.

We also use a gradual warmup schedule, which means that we initially boost our learning rate by a factor decided a priori, our warmup factor equal to 10, and then the learning rate follows our cosine annealing schedule. Basically, we combine the Adam optimization with these procedures in order to reach the objective of smoothing the optimization by following each gradient dimension in the same way and without converging in non-optimal points.

Finally, the last key player of the optimization process is the loss function. For EfficientNet, we tested and adopted a Binary Cross-Entropy Logits Loss function:

$$l_{n,c} = -w_{n,c}[p_c y_{n,c} \cdot \log\left(\sigma\left(x_{n,c}\right)\right) + \left(1 - y_{n,c}\right) \cdot \log\left(1 - \sigma\left(x_{n,c}\right)\right)]$$

Where *c* is the class, *n* is the number of the sample in the batch and *p* is the optional weight for positive examples. As the name suggests this function is a combination of a classical Cross-Entropy Loss with a sigmoid function, which is used as activation before the loss evaluation.

Conclusion: Setting a small batch size of 2, given we do not have many samples, and tuning the hyperparameters with some runs, we reached a score of almost 0.86 (quadratic weighted kappa) with 32 tiles. It is quite good, considering also that we did not perform many epochs (13), given the limited resources at our disposal we could not go further. Furthermore, the tiling approach seems to work pretty well, with an initial score of 0.68, we already got to 0.82 with 12 tiles.

5. Conclusion

We tried to address the problem of prostate cancer classification mainly relying on fine-tuning some of the most famous convolutional networks architectures. We tested different versions and we get a score of 0.86, but we were limited by computational costs. Indeed, we could not be able to run for a long time the VM because we got technical issues. We are pretty confident that with more runs we could get to the 0.90, getting closer to the challenge winners who got around 0.94. We are also confident that using “higher” versions of EfficientNet the results can be improved. The tiling approach proved to be very efficient and allowed to increase a lot the score: the first run of EfficentNet without tiling gets 0.67 of score. Optimizing the space and focusing on the regions with smaller intensity turns out to be an excellent strategy.

There are lot of improvements and tests that could be made. For example, given the small size of the dataset, there is the possibility of testing with full-batch a second order optimization algorithm like L-BFGS, which is a limited-memory quasi-newton method that goes deeper into the data using the Hessian matrix. Furthermore, the integration of masks can be an important resource in order to make a more accurate and region-wise classification.

Anyway, given that this is our first ever task in computer vision, we got good results and we built a guideline for this kind of tasks as we wanted.

Sitography

[1] https://www.cancer.gov/publications/dictionaries/cancer-terms/def/cancer

[2] https://www.cancer.gov/about-cancer/understanding/statistics

[3] https://en.wikipedia.org

[4] https://www.sam-solutions.com/blog/four-best-cloud-deployment-models-you-need-to-know/

[5] https://www.bluepiit.com/blog/different-types-of-cloud-computing-service-models/

[6] https://console.cloud.google.com/home/dashboard?project=panda-thesis&organizationId=0

[7] https://github.com/cs231n/gcloud#set-up-google-cloud-vm-image

[8] https://github.com/Kaggle/kaggle-api

[9] Lecture Collection | Convolutional Neural Networks for Visual Recognition (Spring 2017) Stanford University School of Engineering

Bibliography

- Didactic Material
  - Professor Gribaudo, BABD4_L07 – Cloud computing
  - Professor Ardagna, BABD4_CloudComputing_Ardagna
- Robbins e Cotran, “Patologia generale: Le basi patologiche delle malattie – Vol.2”
- Vivek G Gite, “Linux Shell Scripting Tutorial Ver. 1.0”
- Abraham, Bejoy, and Madhu S. Nair. “Automated Grading of Prostate Cancer Using Convolutional Neural Network and Ordinal Class Classifier.” *Informatics in Medicine Unlocked* 17 (January 1, 2019): 100256.

- Agarwal, Rahul. “End to End Pipeline for Setting up Multiclass Image Classification for Data Scientists.” Medium, June 25, 2020.

- Agrawal, Devanshu, Theodore Papamarkou, and Jacob Hinkle. “Wide Neural Networks with Bottlenecks Are Deep Gaussian Processes.” *ArXiv:2001.00921 [Cs, Stat]*, July 6, 2020.

- Amjad, Rana Ali, and Bernhard C. Geiger. “Learning Representations for Neural Network-Based Classification Using the Information Bottleneck Principle.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, no. 9 (September 1, 2020): 2225–39.

- Anwar, Aqeel. “Difference between AlexNet, VGGNet, ResNet and Inception.” Medium, June 13, 2020.

- OpenAI. “Attacking Machine Learning with Adversarial Examples,” February 24, 2017.

- Bai, Kunlun. “A Comprehensive Introduction to Different Types of Convolutions in Deep Learning.” Medium, February 11, 2019.

- Bergstra, James, and Yoshua Bengio. “Random Search for Hyper-Parameter Optimization,” n.d., 25.

- Brownlee, Jason. “A Gentle Introduction to 1x1 Convolutions to Manage Model Complexity.” *Machine Learning Mastery* (blog), April 28, 2019.

- Buslaev, Alexander, Alex Parinov, Vladimir Iglovikov, Evegene Khvedchenya, and Mikhail Druzhinin. “Albumentations Documentation,” n.d., 78.

- Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs.” *ArXiv:1412.7062 [Cs]*, June 7, 2016.

- “CS231n Convolutional Neural Networks for Visual Recognition.” Accessed August 17, 2020.

- Deis, Alexandra. “Data Augmentation for Deep Learning.” Medium, August 10, 2019.

- Dumoulin, Vincent, and Francesco Visin. “A Guide to Convolution Arithmetic for Deep Learning.” *ArXiv:1603.07285 [Cs, Stat]*, January 11, 2018.

- Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” *ArXiv:1311.2524 [Cs]*, October 22, 2014.

- Glorot, Xavier, and Yoshua Bengio. “Understanding the Difficulty of Training Deep Feedforward Neural Networks,” n.d., 9.

- Goyal, Priya, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour.” *ArXiv:1706.02677 [Cs]*, April 30, 2018.

- Grave, Edouard, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. “Efficient Softmax Approximation for GPUs.” *ArXiv:1609.04309 [Cs]*, June 19, 2017.

- He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. “Mask R-CNN.” *ArXiv:1703.06870 [Cs]*, January 24, 2018.

- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for

Image Recognition.” *ArXiv:1512.03385 [Cs]*, December 10, 2015.

- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Identity Mappings in Deep Residual Networks.” *ArXiv:1603.05027 [Cs]*, July 25, 2016.

- Hoffmann, Mark. “Exploring Stochastic Gradient Descent with Restarts (SGDR).” Medium, November 17, 2017.

- Hu, Jie, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. “Squeeze-and-Excitation Networks.” *ArXiv:1709.01507 [Cs]*, May 16, 2019.

- Huang, Bohao, Daniel Reichman, Leslie M Collins, Kyle Bradbury, and Jordan M Malof. “Tiling and Stitching Segmentation Output for Remote Sensing: Basic Challenges and Recommendations,” n.d., 9.

- *IBM/Image-Preprocessing-for-Deep-Learning-Models*. Jupyter Notebook. 2019. Reprint, International Business Machines, 2020.

- Ioffe, Sergey, and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” n.d., 9.

- Iqbal, Muhammad Shahid, Bin Luo, Tamoor Khan, Rashid Mehmood, and Muhammad Sadiq. “Heterogeneous Transfer Learning Techniques for Machine Learning.” *Iran Journal of Computer Science* 1, no. 1 (April 2018): 31–46.

- Jin, Jonghoon, Aysegul Dundar, and Eugenio Culurciello. “Flattened Convolutional Neural Networks for Feedforward Acceleration.” *ArXiv:1412.5474 [Cs]*, November 20, 2015.

- Kingma, Diederik P., and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” *ArXiv:1412.6980 [Cs]*, January 29, 2017.

- Kostadinov, Simeon. “Understanding Backpropagation Algorithm.” Medium, August 12, 2019.

- Kumar, Niranjan. “Visualizing Convolution Neural Networks Using Pytorch.” Medium, December 17, 2019.

- Li, Zhizhong, and Derek Hoiem. “Learning without Forgetting.” *ArXiv:1606.09282 [Cs, Stat]*, February 14, 2017.

- Lin, Min, Qiang Chen, and Shuicheng Yan. “Network In Network.” *ArXiv:1312.4400 [Cs]*, March 4, 2014.

- Liu, Saifeng, Huaixiu Zheng, Yesu Feng, and Wei Li. “Prostate Cancer Diagnosis Using Deep Learning with 3D Multiparametric MRI.” edited by Samuel G. Armato and Nicholas A. Petrick, 1013428. Orlando, Florida, United States, 2017.

- Loshchilov, Ilya, and Frank Hutter. “Decoupled Weight Decay Regularization.” *ArXiv:1711.05101 [Cs, Math]*, January 4, 2019.

- Loshchilov, Ilya, and Frank Hutter. “SGDR: Stochastic Gradient Descent with Warm

Restarts." *ArXiv:1608.03983 [Cs, Math]*, May 3, 2017.

- Mahendran, Aravindh, and Andrea Vedaldi. "Understanding Deep Image Representations by Inverting Them." *ArXiv:1412.0035 [Cs]*, November 26, 2014.

- Noh, Hyeonwoo, Seunghoon Hong, and Bohyung Han. "Learning Deconvolution Network for Semantic Segmentation." In *2015 IEEE International Conference on Computer Vision (ICCV)*, 1520–28. Santiago, Chile: IEEE, 2015.

- Odena, Augustus, Vincent Dumoulin, and Chris Olah. "Deconvolution and Checkerboard Artifacts." *Distill* 1, no. 10 (October 17, 2016): e3.

- Prakash (Adi), Aaditya. "One by One [1 x 1] Convolution - Counter-Intuitively Useful." Accessed August 13, 2020.

- "PyTorch, Dynamic Computational Graphs and Modular Deep Learning." Medium, accessed June 30, 2020.

- "Pytorch or Tensorflow, Dynamic vs Static Computation Graph." Medium, Accessed June 30, 2020.
- Reddi, Sashank J., Satyen Kale, and Sanjiv Kumar. "On the Convergence of Adam and Beyond," 2018.

- Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, no. 6 (June 1, 2017): 1137–49.

- Saglani, Vatsal. "Multi-Class Image Classification Using CNN over PyTorch, and the Basics of CNN." Medium, April 20, 2020.

- Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." *ArXiv:1801.04381 [Cs]*, March 21, 2019.

- Schelb, Patrick, Simon Kohl, Jan Philipp Radtke, Manuel Wiesenfarth, Philipp Kickingereder, Sebastian Bickelhaupt, Tristan Anselm Kuder, et al. "Classification of Cancer at Prostate MRI: Deep Learning versus Clinical PI-RADS Assessment." *Radiology* 293, no. 3 (October 8, 2019): 607–17.

- Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps." *ArXiv:1312.6034 [Cs]*, April 19, 2014.

- Skorski, Maciej, Alessandro Temperoni, and Martin Theobald. "Revisiting Initialization of Neural Networks." *ArXiv:2004.09506 [Cs, Math, Stat]*, June 4, 2020.

- Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," n.d., 30.

- Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. "Highway Networks."

ArXiv:1505.00387 [Cs], November 3, 2015.

- Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. “Training Very Deep Networks.” *ArXiv:1507.06228 [Cs]*, November 23, 2015.

- Stamoulis, Dimitrios, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. “Single-Path NAS: Device-Aware Efficient ConvNet Design.” *ArXiv:1905.04159 [Cs, Stat]*, May 10, 2019.

- Sutskever, Ilya, James Martens, George Dahl, and Geoffrey Hinton. “On the Importance of Initialization and Momentum in Deep Learning,” n.d., 14.

- Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. “Going Deeper with Convolutions.” *ArXiv:1409.4842 [Cs]*, September 16, 2014.

- Tan, Mingxing, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. “MnasNet: Platform-Aware Neural Architecture Search for Mobile.” *ArXiv:1807.11626 [Cs]*, May 28, 2019.

- Tan, Mingxing, and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” *ArXiv:1905.11946 [Cs, Stat]*, November 22, 2019.

- “The_Organization_of_Behavior-Donald_O._Hebb.Pdf.” Accessed July 14, 2020. http://s-f-walker.org.uk/pubsebooks/pdfs/The_Organization_of_Behavior-Donald_O._Hebb.pdf.
- “TPUs_in_Colab.Ipynb,” n.d.
- Sebastian Ruder. “Transfer Learning - Machine Learning’s Next Frontier,” March 21, 2017.

- Tsang, Sik-Ho. “Review: DeconvNet — Unpooling Layer (Semantic Segmentation).” Medium, March 20, 2019.

- Tsang, Sik-Ho. “Review: Highway Networks — Gating Function To Highway (Image Classification).” Medium, March 20, 2019.

- Wiesler, Simon, and Hermann Ney. “A Convergence Analysis of Log-Linear Training,” n.d., 10.

- Xie, Cihang, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. “Adversarial Examples Improve Image Recognition,” n.d., 10.

- Xie, Saining, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. “Aggregated Residual Transformations for Deep Neural Networks.” *ArXiv:1611.05431 [Cs]*, April 10, 2017.
- Yalniz, I. Zeki, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. “Billion-Scale Semi-Supervised Learning for Image Classification.” *ArXiv:1905.00546 [Cs]*, May 1, 2019.

- Yoo, Sunghwan, Isha Gujrathi, Masoom A. Haider, and Farzad Khalvati. “Prostate Cancer Detection Using Deep Convolutional Neural Networks.” *Scientific Reports* 9, no. 1 (December 20, 2019): 19518.

- Yuan, Yixuan, Wenjian Qin, Mark Buyyounouski, Bulat Ibragimov, Steve Hancock, Bin Han, and Lei Xing. “Prostate Cancer Classification with Multiparametric MRI Transfer Learning Model.” *Medical Physics* 46, no. 2 (February 2019): 756–65.

- Zeiler, Matthew D., and Rob Fergus. “Visualizing and Understanding Convolutional Networks.” In *Computer Vision – ECCV 2014*, edited by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, 8689:818–33. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014.

- Zoph, Barret, and Quoc V. Le. “Neural Architecture Search with Reinforcement Learning.” *ArXiv:1611.01578 [Cs]*, February 15, 2017.

- Zoph, Barret, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. “Learning Transferable Architectures for Scalable Image Recognition.” *ArXiv:1707.07012 [Cs, Stat]*, April 11, 2018.