Deep Learning Methods for Processing Digitized Herbarium Specimens

Filipe Martins[1][97418]

Instituto Superior Técnico, Universidade de Lisboa
`filipe.seleiro.martins@tecnico.ulisboa.pt`

Abstract. We will go over a solution to solve the problem faced in multiple Natural History Museums and similar institution of cataloging extensive herbarium collection we will research a model using state of the art Deep Learning models .

Keywords: Preceptron · MLP · CNN · RNN · Transformer · Computer Vision · Object Recognition · Region Segmentation · Text Recognition · Image Captioning · Optical Character Recognition · YOLOV4 · Swin · SPAN · ViT · CVT · CPTR · SPP · ResNet · CSPNet · Attention Module · Self-Attention .

Table of Contents

1 Introduction 3
2 Fundamental Concepts 3
  2.1 The Perceptron and MLP 3
  2.2 Training the MLP 6
  2.3 Recurrent Neural Networks (RNN) 8
  2.4 Convolutional Neural Networks for Computer Vision 12
  2.5 Self Attention - The Transformer Architecture 17
3 Related Work 20
  3.1 Object Recognition and Region Segmentation 20
  3.2 Image Captioning 29
  3.3 Text Recognition in Image Data 33
  3.4 Overview 35
4 Research Proposal 36
  4.1 Thesis Statement 36
  4.2 Datasets and Evaluation Methodology 37
  4.3 Overview on Proposed Method 37
5 Planning for Next Semester 39

Appendices 43

1 Introduction

Hundreds of herbarium collections, currently in Natural History Museums and other similar institutions, have accumulated a valuable heritage and knowledge of plants over several centuries. Recent initiatives started ambitious preservation plans to digitize this information and make it available to botanists and the general public through web portals. Such information is crucial for the study of plant diversity, ecology, evolution, and genetics. Herbarium is a collection of preserved plant specimens and meta data used for scientific study. These are preserved using a technique called "exsiccatae" that consists mounting in a sheet of paper dried from herbaria. Even though the method of digitization and cataloging using computer vision, as well as the machine learning approaches applied to herbarium sheets, can both be considered promising, recent methods based on deep neural networks are still not well studied in this particular problem domain in comparison to other areas. We will go over some model that can be used to achieve next generation precision and utilities for this field of cataloging Herbarium. To achieve this goal we will explore the new techniques and new models that represent the State of the Art in certain application that can be useful to our goal.

2 Fundamental Concepts

Computational Intelligence is defined by the combination of techniques through design and application development, usually inspired by biology and nature for the purpose of giving a computer the ability to learn a specific task, defined as nature-inspired computational methodologies. This section consists of concepts required to understand the methodologies and techniques for the remaining part of the report. We will cover the basis of what a neural network is, as well as the thoughts and inspiration behind it. We'll also be analysing how the machine can learn, train and optimize said networks. Later in this section, we'll also be giving a brief overview over the uses of convoluted neural networks applied to computer vision and some of the thought process behind it. Finally, we'll be going through some newer methods that are starting to be used in computer vision with some great promise.

2.1 The Perceptron and MLP

Similar to nature, computed neural networks have a neuron called perceptron which in is simplest form represents a value . This node (perceptron) which constitutes an activation value that is calculated through a series of linear equations. In similarity to our neurons, they receive synapses from outside sources or other neurons, and get activated by the same principal. The activation value is calculated through the weighted sum of multiple inputs, also called features, which mimics the way the neurons react to multiple stimulus. To add some degree of freedom to this model, a bias value is added that represents the minimum state in which the perceptron is activated. This value is not related to any of its inputs.

Due to the perceptron limitation, it cannot solve problems that aren't linearly separable since the equation: activation value $= \sum_{n=1}^{n} x_n.w_n + bias$ defines a hyperplane. To further enhance this model for non-linearity problem solving, it's added a nonlinear function called activation function (also known as nonlinearities). This function mimics the neuron firing rate while the previous function represents the amplitude of those pulses. The output of the perceptron is calculated through the activation function. Some of the Common activation functions used are Rectified Linear Activation (ReLU) ,Logistic (Sigmoid), Hyperbolic Tangent (Tanh) [1] .

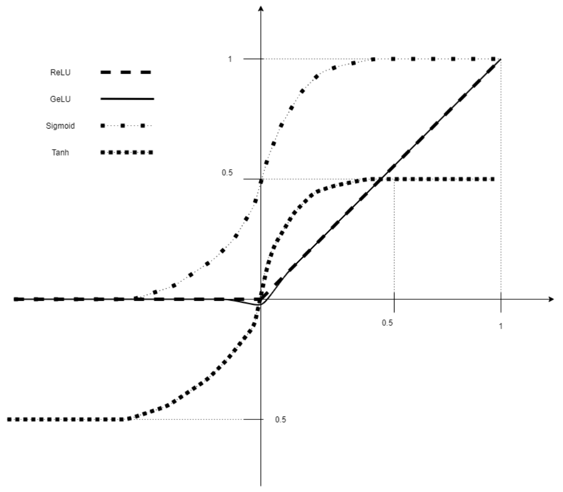

Fig. 1. Visualization of Some Activation Functions.

$$\text{node value} = f(x) \text{ were } \begin{cases} \text{x = activation value } \leftarrow \sum_{n=1}^{n} x_n.w_n + bias, \\ \text{f = activation function.} \end{cases} \quad (1)$$

It's crucial to point out that this function has some peculiar characteristics, one of them being the function cannot be linear around the value used for decision making, which is usually around zero. This function must have a possible derivative and it's not a constant throughout its domain.

[1] For future reference Softmax activation function is a variation on sigmoid function both are limited between a value between 0 and 1, ReLU is computationally very light to apply but as a characteristic of this function it removes negative values by setting them value 0 also has non derivative in 0. Gelu is an alternative to ReLU and solves the problem of the derivatives.

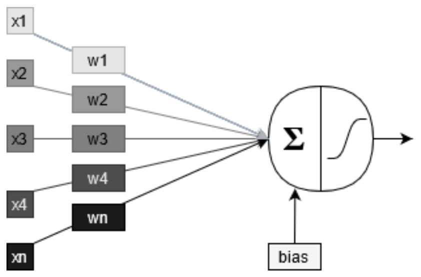

Fig. 2. Perceptron.

To enhance the capabilities of the perceptron to achieve non trivial solutions, the concept of Multi-Layer Perceptron (MLP) was created. Usually, it's used with 3 layers: an input that represents the values of each feature, a hidden layer where the first set of perceptrons resides, and one output. The latter is simply another layer of perceptrons that gives the desired output values. This standard structure is enough to solve any problems with the drawbacks on pre-processing complexity [12].

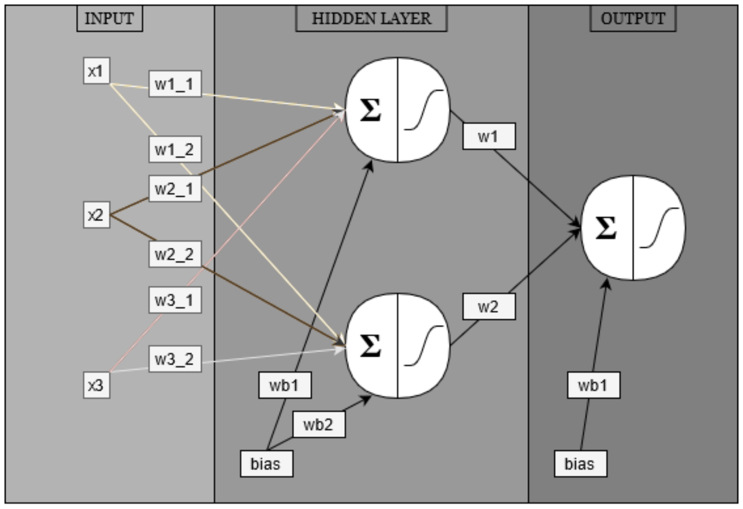

Fig. 3. Feed Forward NN / MLP.

There are multiple ways to connect each output to an input. In the example shown above3, since every connection is in series and each layer is connected to the next in every combination possible, this particular MLP is arranged as a Feed Forward Neural Network.

2.2 Training the MLP

There are multiple algorithms and paradigms that can provide learning capabilities to the MLP. To keep this short, we'll just be going in depth into the method mentioned in this report. The supervised learning usually starts with a training set that has X examples and the expected results, Y Labels, and the goal is to make the MLP return values closest/identical to the expected values, in order to minimise the error on the output for a giving input. This is usually done by back propagating the error while optimizing the parameters until it achieves a minimal Loss. In order to figure out how to represent perception of knowledge on a perceptron, the two most common ways to do it is through two mathematical formulas: the euclidean distance (minimizing problem) and the internal product (maximizing problem). The primary characteristic of these formulas is that they can correlate the similarity of two values (in this case between the expected value and the actual output) for a given input, therefore we have the value of the error. Giving the example of the MLP's structure demonstrated in figure 3, once the generalization of number of inputs (i) is applied, the number of perceptrons on the hidden layer (j), and the number of outputs(k), the formulation of the output calculated by definition is:

$$\mathrm{Out}_k = \sum_j w_{jw} - \left(\sum_i w_{ij} - X_i \right) \tag{2}$$

The goal is to minimize the error given by the difference between the output and the expected values so we can formulate a loss function (L).

$$\mathrm{L} = \frac{1}{2i} \sum_i (||Y_i - Out_{X_i}||)^2 \tag{3}$$

The current method to make the neural network learn, and therefore optimizing its parameters to a learning set, is using the chain rule (also known as back-propagation) combined with the gradient descend in an algorithm that ensures the parameters have been optimized by trending to a local minimum with the intent of minimizing overall loss. Applying the chain rule each step of the loss function(L) can be expressed as a multiplication of several derivatives of each step of the process of back-propagation. Calculating the derivative of the error by each component tells us the direction the gradient descend must trend in order to minimise the loss by each component.

The problem of minimizing loss for one example set is over-fitting our neural network to a specific example set and not for the general problem. Fortunately, there is also a parameter that tries to prevent this from happening, as it defines learning rate, which translates the maximum variation applied to each parameter when updated. Optimally, this parameter should decay over the learning process, however the current method of single batch gradient descent is not optimal since the learning process is only one big single process. To address this problem, there is an optimization that can be used called iterative gradient descent or Stochastic gradient descent (SGD). Since the gradient descent is a single process

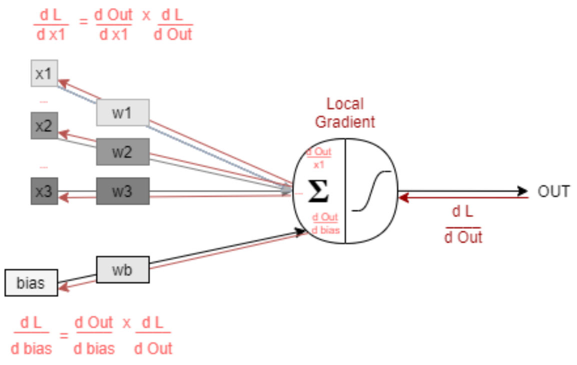
d L / d x1 = d Out / d x1 x d L / d Out
x1
x2
x3
w1
w2
w3
Local Gradient
d Out / x1
Σ
d Out / d bias
OUT
d L / d Out
bias
wb
d L / d bias = d Out / d bias x d L / d Out

Fig. 4. Back propagation applied to a Perceptron.

that looks for the outcome of the whole training batch in order to optimize every parameter by using mini-batches, it allows for the optimal setting in order to make the learning rate more dynamic, since we divide our training set in multiple batches while updating the parameters multiple times. Furthermore, by allowing this to happen, it allows for some fluctuation in each update (since every batch is different), which then ensures that the function jumps to a better local minima. This method is common practice because it's easier to compute and allows for better results.

To solve the problem of over-fitting there was created a method that solves this problem by either by randomly or intentional dropping some nodes in order to add some variance/noise for the network surpass some local minima or vanishing gradients [23].

There are some challenges presented to this point; one is knowing when to update the learning rate/what value is optimal, and the other is ”should every parameter have the same learning rate?”. This is usually solved by applying one of multiple known algorithms. Momentum [20] is a method that improves the speed of SGD, which accelerates the global minima. It achieves this by applying momentum concept in physics and accelerating to the current trending values, therefore allowing it to escape a local minima thanks to its ability to keep momentum figure 5.

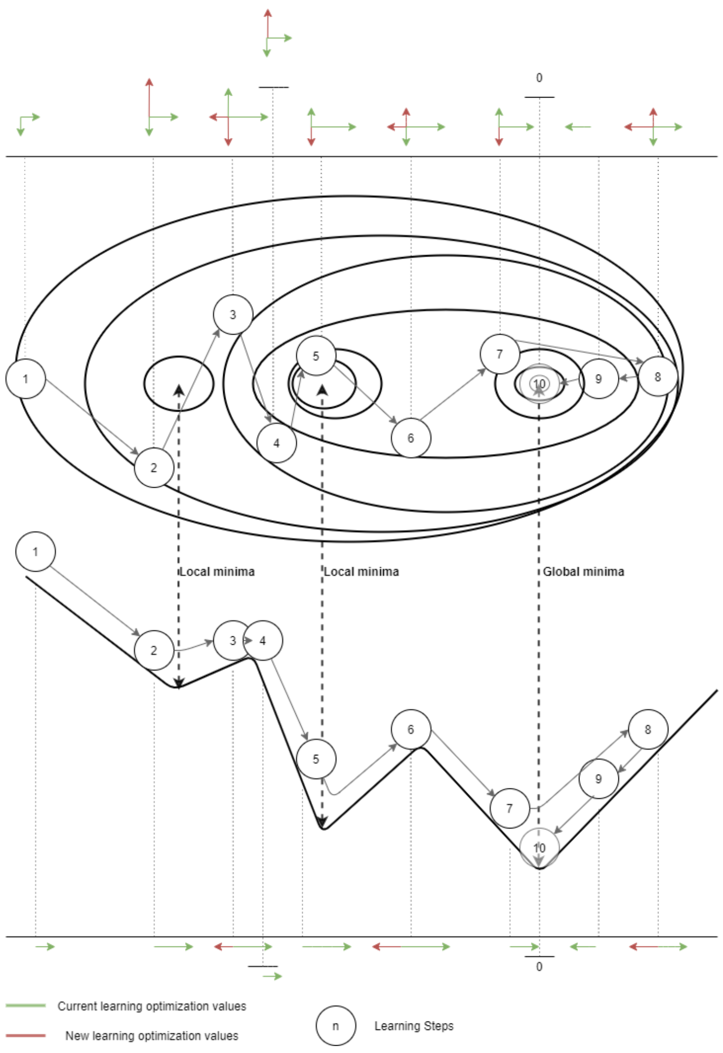
0
1
2
3
4
5
6
7
8
9
10
Local minima
Local minima
Global minima
0
Current learning optimization values
New learning optimization values
n
Learning Steps

Fig. 5. Momentum visualization.

2.3 Recurrent Neural Networks (RNN)

The recurrent neural network (RNN) as we know it was developed based on the first implementation of Hopfield Networks [11] that had been developed to serve as associative memory. The way this model is constructed has each neuron represented by a binary threshold. So this works by; when the memory has a recall of similar pattern (previously trained/imprinted), the output will converge to those learnt values, but when the input has no similarity, it will start to diverge into other random values. To achieve this each neuron is connected is almost as feedback loop, but each output of one neuron isn't connected to itself but instead to every other, therefore no self-feedback is presented.This structure is similar as other devices that serve a similar propose like the electrical device Flip-Flop that is used to store a single bit value.

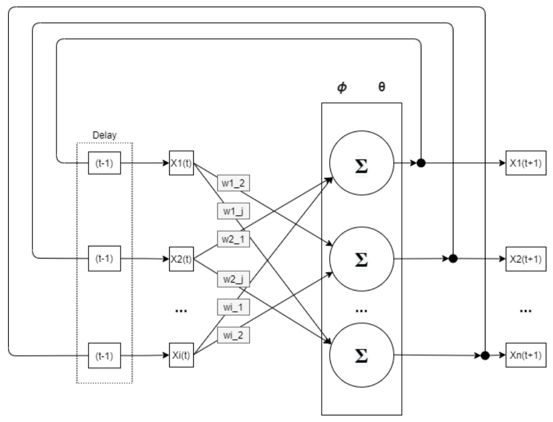

Fig. 6. Hopfield Network.

Inference is the process of calculating the predicted value, and since this network works in binary, there is only two possible values; high and low. The value is calculated through:

$$\mathrm{s}_i \leftarrow \begin{cases} +1 \text{ if } \sum_j w_{\text{i\_j}} s_j >= \theta_i, \\ \text{-1 otherwise.} \end{cases} \tag{4}$$

Where s represents state (pattern) and θ is the threshold for activation of our neuron. This concludes that the value high is only presented when the sum of the neuron inputs exceeds the threshold value. To make this Hopfield network learn, it's used the Hebbian learning method. The theory is the weight of connected neurons has a higher positive value when two connected nodes tend to have the same value, either by being both positive or both negative at the same time, while those that tend to be opposite have a strong negative weight. Can be expressed by:

$$w_{\text{i\_j}} = \frac{1}{n} \sum_{x=1}^{n} \epsilon_i^x \epsilon_j^x \tag{5}$$

Where $w_{\text{i\_j}}$ represents weight of the connection from neuron j to neuron i, n is the number of training patterns, and ϵ_i^x the ϵ_j^x input for neuron i. This is learning by single epoch (weights updated after all the training examples are presented). Since there is no self-feedback, the value of the weight $w_{\text{i\_j}}$ when $i = j$ is set to 0. There was a need to improve this concept since Hopfield Networks only works with expected values and not with experience, thus it can't generalize. As such to overcome this limitation, it was developed a RNN with Non-Stationary Inputs. One of the first successful models was the Elman and Jordan Networks. These network architectures are similar to the base MLP Feed forward; they both have an input layer, a hidden layer and an output layer. The exception is the hidden layer also outputs and receives the input from a parallel hidden state layer of the same size for the Elman network, while in the Jordan network, the hidden state layer receives the input from the output stage.

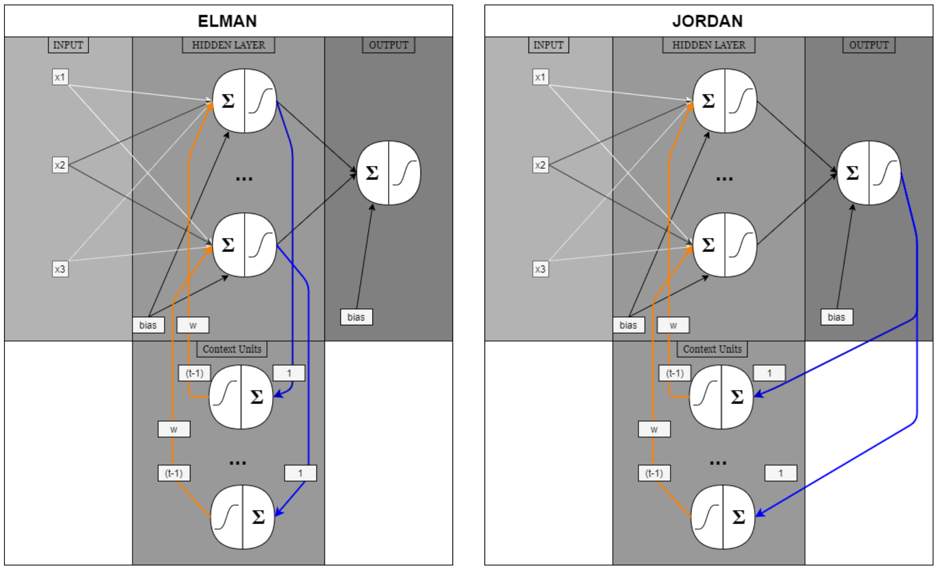

Fig. 7. Jordan and Elman Recurrent Neural Networks Architectures.

The output is calculated by looping the knowledge to our context layer so the output can variate over time ,this iterations are required to it memorise some patterns.

$$\mathrm{Out}_t = \sigma_i(W_i.h_t + b_y) \tag{6}$$

Where h_t is defined over lopped iterations. $h_t = \sigma_h(W_h x_t + Wc_h h_{t-1} + b_h)$ and x_t is the input vector, h_t represents the hidden layer and Out_t is the output vector. The trainable values are represented by: W representing weights relative to the input and Wc the context layer weights and every present bias values where a connection is established. The activation function is represented by σ. To train this network, the method used is similar to a normal MLP; it's done through back propagation although there are some problems that are inherit of this architecture. The first problem is that there are some limitations of what activation function we can use, its known by experimentation that there is a problem with convergence when using the Tanh and ReLU as activation functions because the values on the hidden state layer get easily overwritten. When training this network since we do multiple passes through the hidden state layer the gradient descend can either ”blow up” or vanish there for making hard to get fit weights. The Long Short Term Memory unit (LSTM), was created to solve previous attempts limitations on keeping some memory for a generalized input. Part of the solution was the creation of the LSTM unit in contrast of the perceptron it adds multiple gates that allow some contextualized information to be kept or forgotten for the inference process [10].

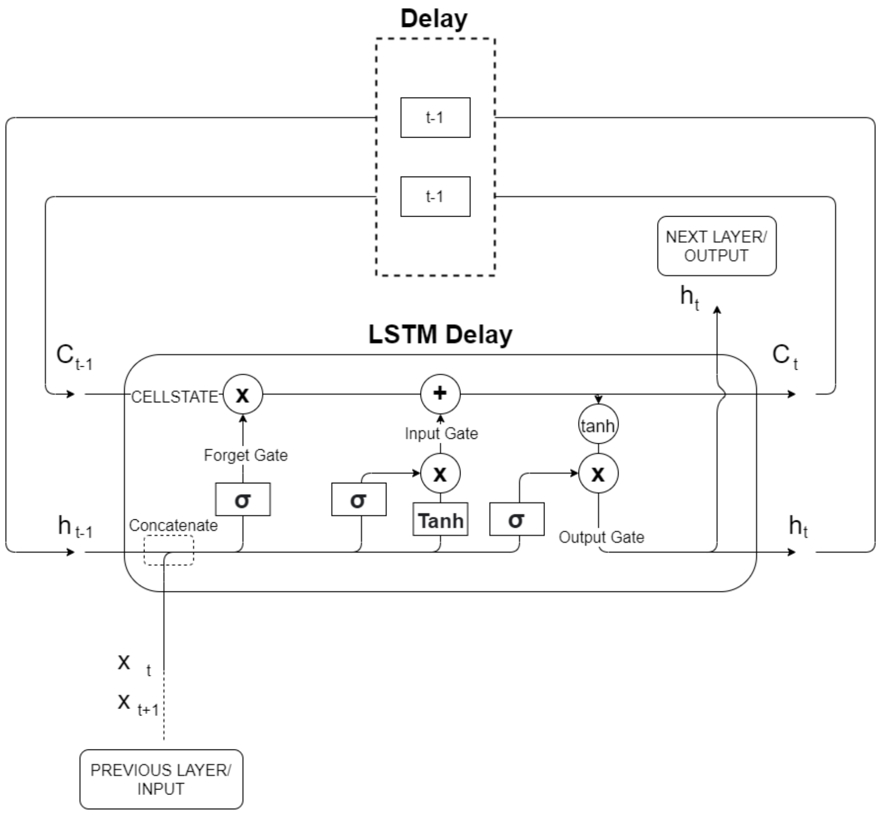

Fig. 8. LSTM Unit and Architecture.

According to our figure 8 we will explain some concepts that came along with this new architecture.A Cell State represents a value that is kept between iterations of the unit this state can be modified through two gates: the forget gate (f_t)and the input gate (i_t).

$$c_t = f_t * C_{t-1} + i_t \tag{7}$$

Forget Gate is the gate that is responsible for what information is kept, this gate is modulated by a sigmoid function that means the value is modulated between 0 and 1 so when the value is nearest to 0 it means for the cell state need to forget some contextual information.

$$\mathrm{f}_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{8}$$

The input gate is responsible for what information is added to the new current cell state this is done by a adding operation our input gate state and the cell state. To add some degree of freedom this information is contextualized between two functions one is a tanH that simply represents the inference value and a sigmoid that serves to calculate the weight of such inference value.

$$\mathrm{i}_t = \sigma(W_\sigma.[h_{t-1}, x_t] + b_\sigma)\ .\ tanh(W_{tanh}.[h_{t-1}, x_t] + b_{tanh}) \tag{9}$$

The output is given by a modulation of the cell state to the current input therefor the output information is almost always based on the past.

$$\mathrm{h}_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) * tanh(C_t) \tag{10}$$

The LSTM popularity is connected to the good results in multiple applications, this network is capable of utilizing over 100 backwards iteration thus proving to be very useful on application that is require past context. The training of network with LSTM layer(s) is done through back propagation of the error adjusting the parameters like a more traditional MLP.

2.4 Convolutional Neural Networks for Computer Vision

The concept on Convolutional Neural Networks (CNN) was specially developed for imaging processing. They were developed based on the working of the neurons of an animal visual cortex, individual cortical neurons are only connected to a certain region of the visual field known as receptive field. The receptive fields of different neurons can partially overlap to ensure they cover the entire visual field. For simplification we will use examples applied to computer vision. Starting with how a basic architecture is usually constructed for these type of problem, then we'll go in depth into each layer and what their functions are, as well as some concerns to be on the look out. For a CNN to make any decision over an image, we have to achieve good results in two major steps: feature extraction and classification. For that to happen, the user must be concise on the feature selection relevant to the problem, together with choosing a proper method for classification. The feature extraction process is everything related with the preparation of the input image; the name of the modified image is known as feature maps. Classification is usually done by dense network (MLP) that makes the decision over the previous feature maps. Since the CNN is based on animal vision, the input by established standards in the feature set has the name NHWC, making each letter correspond to a video-like component where N is the number of images in the batch, H the height of the image, W the width

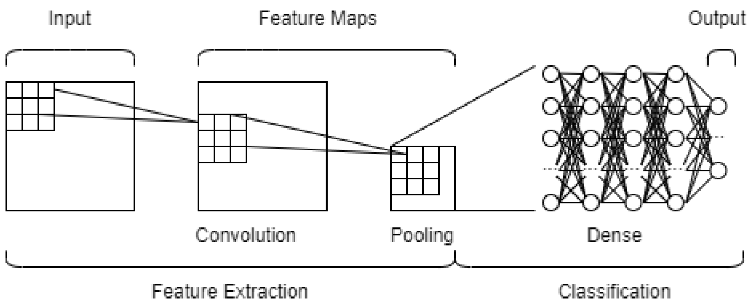

Fig. 9. Basic CNN architecture.

of the image and C the number of channels of the image (ex: 3 for RGB, 1 for grayscale).

Before talking about the operation convolution, we need to understand what are the existent types of convolution matrix and how they work. The utility of the convolution matrix (kernel) serves to do feature selection by removing unnecessary information from our image. The Kernel represents a matrix of coefficients usually with the common size of 3x3 or 5x5, that is applied to a patch of pixels in the process called convolution. By doing this we can extract some features of the image that are relevant to our model, like edges detecting or applying some sharpening, blur and other pre-processing that seems relevant to our problem.

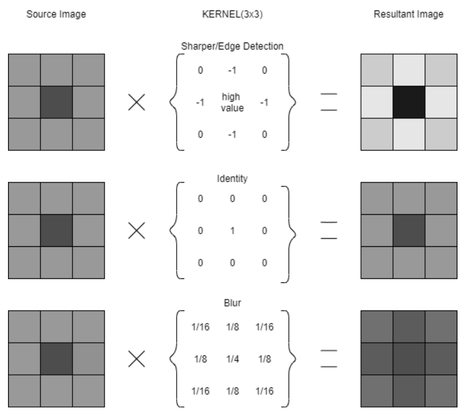

Fig. 10. Kernel visualization.

The Convolution is a process of applying our kernel filter to our image for the purpose of filtering the relevant data for our model. By definition, we can represent the convolution by expression:

$$g(x,y) = k * f(x,y) = \sum_{dx=-i}^{i} \sum_{dy=-j}^{j} w(dx,dy).f(x+dx,y+dy) \qquad (11)$$

Where g(x,y) resultant image, k is the kernel and f(x,y) our original image. This generalization is only applied when the resultant image is the same size as the original, due to padding or striding values the resulting feature map may not have the same size when compared to the original.

The Dilated Convolutions is a special case of the conventional convolution whose inspiration of its development is to increase the receptive field of the operation. By increasing the receptive field, it allows for aggregation of multi-scale contextual information on the rest of the image allowing these techniques to happen without losing resolution [29]. To increase the receptive field with the Conventional convolution, we need to increase the kernel size thus losing border information (described in depth on the next paragraph) to prevent having to add padding.

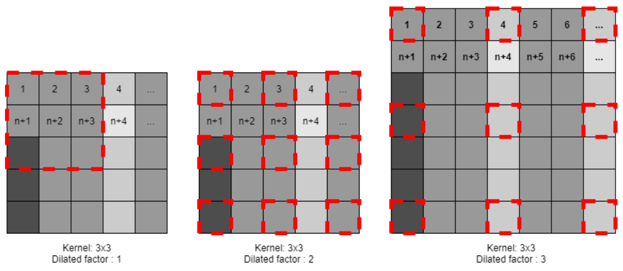

Fig. 11. Dilated Convolutions Visualization.

Padding is a technique that consists on filling the boundaries of the image with a value (usually zero), in order to prevent loss of border information when applying convolution. Since the kernel matrix is usually bigger than 1x1, the outer edge pixels do not count for the center of the cross correlation, resulting in some loss of information. This technique also serves to keep the same image size in order to simplify the code. This case is exemplified by the figure 12 below.

Stride is a value that represents the number on indexes the kernel jumps for the next convolution. It can also be responsible for some resolution loss. This value should be very carefully chosen in conjunction with padding and the kernel size due to some impossible combinations that result in part of the convolution occurring outside of the picture.

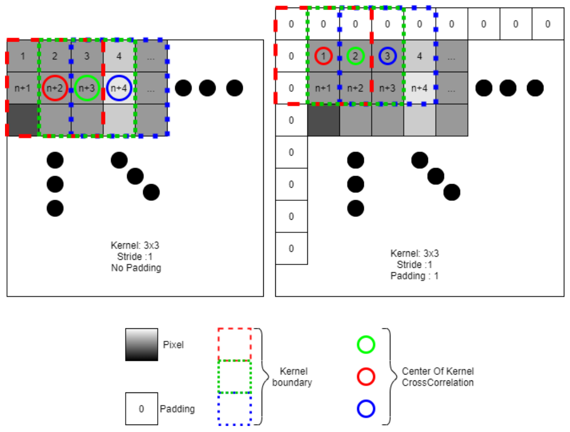

Fig. 12. Padding Visualization.

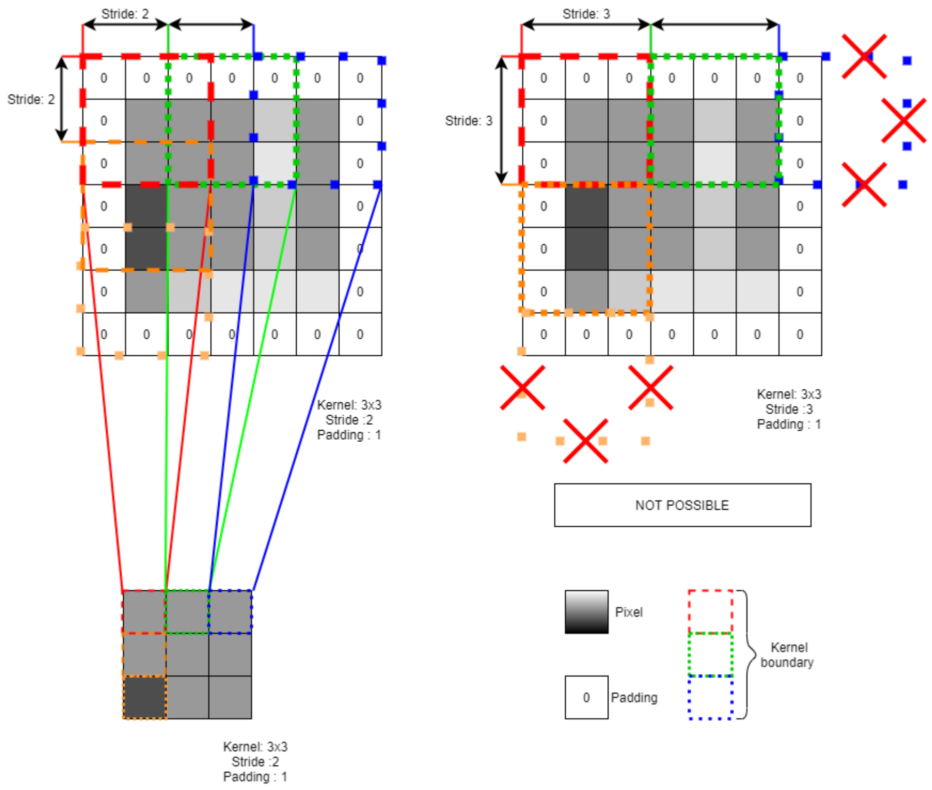

Fig. 13. Stride visualization.

To calculate the size in one of the dimensions of the resulting feature map, we need to take into account some of the parameters explained earlier. When this formulation is applied, the result must be integer, otherwise it's impossible for the chosen values of Stride, Padding and Kernel size to be compatible, thus resulting in the convolution happening outside our image boundaries.

$$\text{FeatureMap}_w = \left(\frac{W - K + 2P}{S}\right) + 1 \tag{12}$$

W represents the input size in one dimension ,K is the Kernel size ,P is the padding and S is the stride.

Pooling operation is used to reduce the spatial dimension, resulting in a down sampling of the feature map. This operation affects the weight and height, but not the depth. To apply this logic of down sampling there are multiple methods to convert a patch of pixels into a single output, the most common being: maximizing pooling, where the max value of the patch is the only value kept; and the average, where it's kept the context of the whole patch.

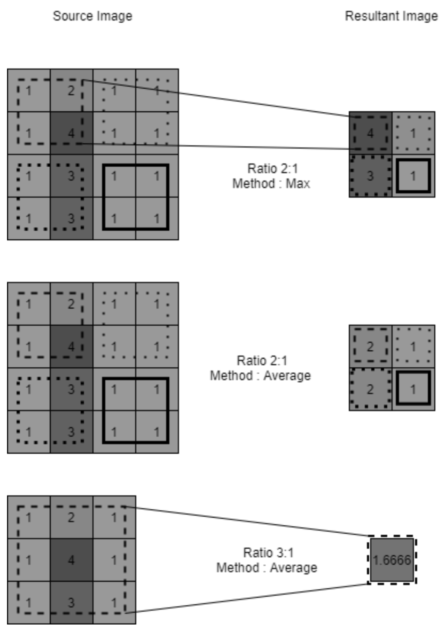

Fig. 14. Pooling visualization.

There are two main implementations that are used for image classification: one is the Linear Classifier; another is the Dense Neural Network (DNN) Classifier. There are advantages and disadvantages for both implementations; Linear

Classifier is usually much faster and can be accurate when talking to simple problems that can be easily linear and separable in data format; while the DNN is more complex and computationally expensive,it is more suitable for general purpose, for the ability of better distinguishing scattered data sets. Linear Classifier achieves its predicaments by applying a linear combination of the feature set applied. Thus this model can only distinguish multiple classes that can be separated by a hyperplane. This method is extremely light to compute. DNN Classifier implements a classical version of MLP as we described before. The standard architecture is as a Feed forward Neural Network as is a common practice since is the most general purpose Classifier due to the flexibility of the MLP to separate disperse clusters of classes.

2.5 Self Attention - The Transformer Architecture

Despite the multiple advances on this field of computer vision, there is a new emerging architecture that was developed specially to be used with Natural Language, in the field of natural language processing (NLP) [24], the Transformer networks also known as self attention networks. Transformer networks have been developed by google, and multiple models based on this architecture are a huge success like BPT and Bert achieving good results for the NLP [25]. There was a need to experiment and expand the use cases of this architecture with some results already proven to be promising for the field of computer vision [6]. The main architectural advantage is that the Transformer networks are highly parallel process. Our main goal is to predict accurate results based on the context of the input. There are two main processes to this type of architecture that implements encoder-decoder structure. One is the encoder where the input sequence $x = (x_1, x_2, ..., x_n)$ is going to get encoded (conversion to tensor) to a continuous representation on our space with vector of $z = (z_1, z_2, ..., z_n)$, then for a given z its passed to a decoder that generated a vector of our output $y = (y_1, y_2, ..., y_n)$ thus this architecture been tensor to tensor.

Before talking about our encoder stack, there is some pre-processing required for our transformer to work as intended. This process begins with the *embeddings*, which consists encoding our input to a vector of size d_{model}. This size is static for any input and it is going to have a representation in our vector space, optimally tokens with similar meaning should be closer to one another in our dimension. Since this architecture doesn't contain any type of recurrence it is necessary to add positional context for each token, this is done through with positional encoder. This module simply takes the size of our token and modulates through a sinusoidal function with different frequencies according to the size.

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{/textmodel}}}\right) \tag{13}$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{/textmodel}}}\right) \tag{14}$$

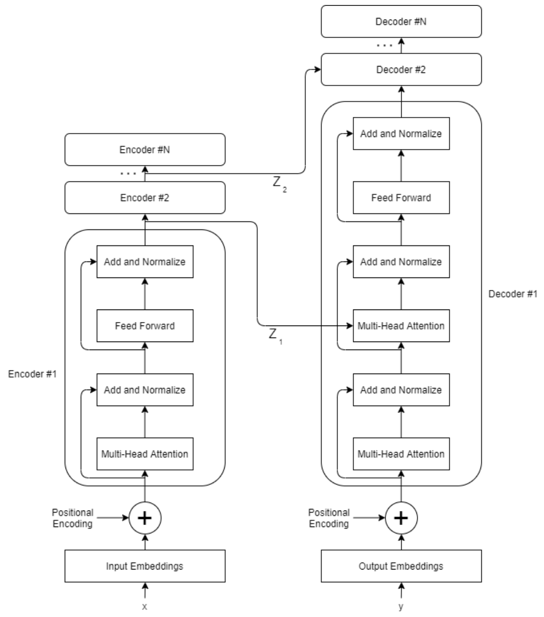

Fig. 15. The Transformer neural network architecture.

Where *pos* is the current position and i is the total dimension.

An encoder module is constructed mainly with two sub-layers. The first one is a multi-head attention mechanism that focuses on how surrounding positions affect the current one, it achieves this by doing a all to all comparison and calculating a attention value. The second one is a simple position-wise fully connected feed-forward network. Decoder structure is similar to the encoder blocks while the difference is the repetition of two sub-layers on each decoder block. The additional sub-layers that composed the decoder block are a normalizing layer that regulates the input to an additional multi-head attention layer. This additional layer does a process called masking with the propose to impose that the prediction over the position i are only imposed and related by less than i positions. Where this network improves is the new mechanism imposed on the multi-head self attention layers allowing to make a correlation between the inputs and an output set based on the best probability of our inference. The way the functions achieves this is by calculating a attention value for each input that can be described Scaled dot product of multiple values. Those values that compose the aforementioned attention function are defined by mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.

The scaled dot-product attention as described in the research paper 16, is calculated for each embedding where each will generate a query vector and a set of key-value vectors. Those vectors are a representation of useful abstractions

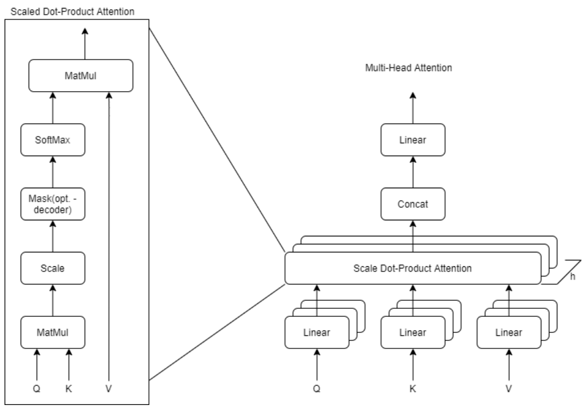

Fig. 16. Scaled-Dot product and Multi head-Attention.

relevant to our problem. The vector are generated by multiplying our input vector and a weight of matrices that was previously trained both Query(Q) and Key(K) vector have the same dimension d_k and d_v for the Value vector. Since in practice a set of queries are computed simultaneously, we packed them together into a matrix ($Q \in \mathbb{R}^{w \times d_k}$), that consists a matrix with the queries vectors multiplied by the weight matrix ($W^Q \in \mathbb{R}^{model \times d_k}$). The same process is applied to keys vectors and values vectors where each is multiplied by a two weight matrices ($W^K \in \mathbb{R}^{model \times d_k}$ and $W^V \in \mathbb{R}^{model \times d_v}$) and also packed together into matrices: ($K \in \mathbb{R}^{w \times d_k}$) and ($V \in \mathbb{R}^{w \times d_v}$). Our attention value will be represented as:

$$\text{Attention}\left(Q, K, V\right) = softmax\left(\frac{QK^t}{\sqrt{d_k}}\right)V \tag{15}$$

The scale factor was needed to counteract where large values produced by the dot-product could push the Softmax function to regions where gradient is small $1 \ / \ \sqrt{d_k}$. So when applying scaling, our large values number will be pushed closer to zero where the Softmax function has more variance thus improving on this limitation. The multi-head attention mechanism allows that our model can access to multiple representations in different sub-spaced by concatenating the information from multiple attention layers, also known as heads. This can be expressed as:

$$\text{MultiHead}\left(Q, K, V\right) = Concat\left(head_1, head_2, ..., head_h\right)W^O \tag{16}$$

$$\text{Where } head_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

The training of this network surprisingly it's done by Back-propagation of the error and there for optimizing weights. Now with the overview of the architecture complete, it's important to mention that this model does not have any context over the input information and how it's related to one another. So it is necessary to add some information how each input is related.

3 Related Work

In this chapter, we'll go over implementations and models that have been researched which can be used on our problem and explain the State of The Art (SoTA) on some of the components and techniques used. We will need to separate each component and each model that can fulfill those tasks in order to give a brief overview of the research and implementations used on each paper. Starting with Region segmentation, we will need to identify and separate each of the components from our image that is the herbaria sheet. These components can be: the written text fields and images of the specimen to retrieve information and associate to our herbarium sheets for cataloging purposes. With the separation done, we'll need to use a Image Captioning model, or as a backup model we can use also Optical Character Recognition (OCR) + NLP(for field recognition) over the region of the image where text data is located and identifying each field required for cataloging. The last model we will aim to use is a Image Captioning to add a brief descriptive caption of the specimen. It is also worth to mention that there's a paper published with the same goal that is only focused on the region of the image where the text label is located using Optical Character Recognition (OCR) to retrieve the information from said label, the paper name is : *"Towards a scientific workflow featuring Natural Language Processing for the digitisation of natural history collections"* [19]. Although this paper has the same goal, our methods will be different and will be used as a performance reference.

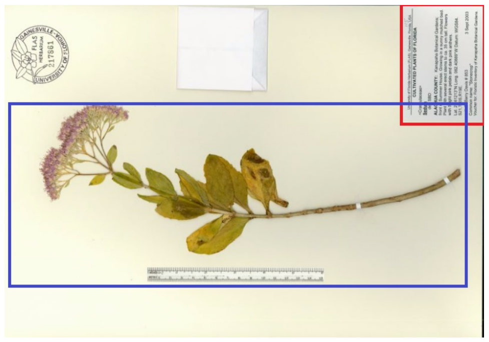

Fig. 17. Demonstration of the Herbarium-Sheet.(IMAGE BY FLORIDA MUSEUM & UNIVERSITY OF FLORIDA HERBARIUM)

3.1 Object Recognition and Region Segmentation

The state of the art (SoTA) in this field of Object Detection mostly represented by Convolutions Neural Networks (CNN), with a honorable mention of

[14] AlexNet (with a innovating methodologies) and the recently Transformers neural networks. The way we are going to use Object Recognition and Region Segmentation is that we need to identify what portions of the input image that corresponds to text where our meta data resides (Name of the herbaria, local of capture, date, etc.), and the image of the herbaria specimen, both for later processing. The Models that will be used as reference model for CNN in the field of image segmentation will be the model of YOLO (You Only Look Once)v4 [1] and a SoTA Transformers Neural Network called Swin [18], which has been the top chart in accuracy Object Detection on COCO data set.

The SOTA in the field of object detection Networks using CNN architectures can be separated into two main categories: One-Stage detector, where we will explain the implementation of YOLO; and Two-Stage detector, with the explanation of the second stage of R-CNN. Before explaining the proposed implementations of the models YOLOV4, we need to clarify some methods and the basic structure for this type of purpose.

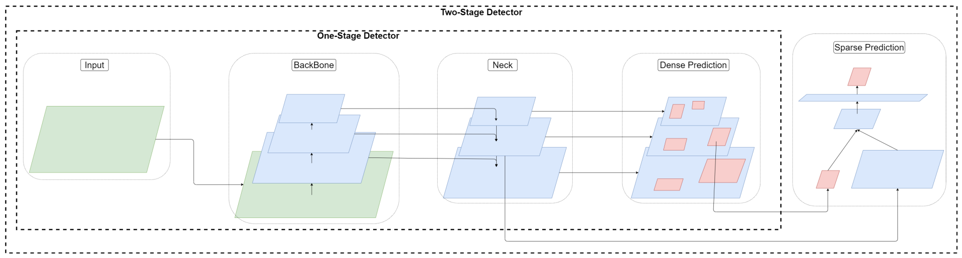

Fig. 18. State of the Art typical CNN Architecture for object detection.

Each block of the architecture has a function: the Backbone where the feature-extraction occurs, the Neck block which purpose is to add extra layers and some refinement between the backbone dense prediction block, and the Head that can be single stage or two stage and is usually where the classification occur and the attribution of bounding boxes.
Before explaining the YOLOV4 architecture we will go over some newer techniques.

The backbone used on YOLOV4 is the CSPDarknet53 that applies the concept of Cross Stage Partial Network (CSPNet) [26]. The goal of technique is to lighten the weight of the computational need and memory costs by achieving richer gradients. This concept works because having richer gradients means there is a reduction of needed larger digits representation than our native processor can achieve (e.g Less bits alleviates memory requirements so naturally achieving faster calculations, also Double Float Precision calculations need more Processor Clocks to calculate translating on slower calculations). The way this technique achieves richer gradients is by partitioning feature map of the base layer into two parts and then merging them through a proposed cross-stage hierarchy.

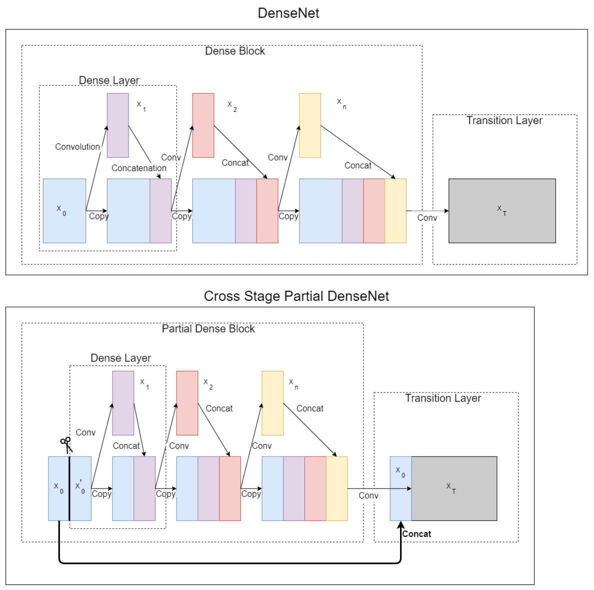

Fig. 19. Cross Stage Partial Network applied to DenseNET.

Another concept that is used besides the CSPNet in CSPDarknet53 for YOLOV4 is that the original concept of Darknet53 [21] actually is a hybrid network with CNN and residual neural network (ResNet) [8]. The principle is similar to the CSP where the information can skip layers and still be processed together with features like relation $F(x) + X$ in order to prevent vanishing gradients. It also serves to mitigate a problem of accuracy degradation; this happens when there is a excess of layers which then leads the deep neural network produces to higher training errors. The hypothesis created is that its easier to optimize residual mapping than to optimize an unreferenced mapping.

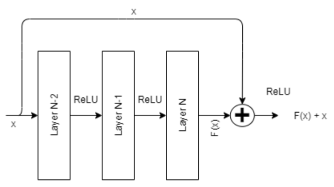

Fig. 20. ResNet exemplification .

The YOLO and R-CNN networks has at is disposal multiple techniques that enhance the receptive field, one of the used techniques is the use of Spatial

Pyramid Pooling (SPP)[9]. This technique was created to eliminate a technical limitation of the network and requires fixed image sizes, plus the SPP ability to generate a fixed length output regardless of the input, therefore avoiding cropping or distorting techniques where we can modify or lose context to our convolutions. The SPP layer pools the features and aggregates into a single input the information.

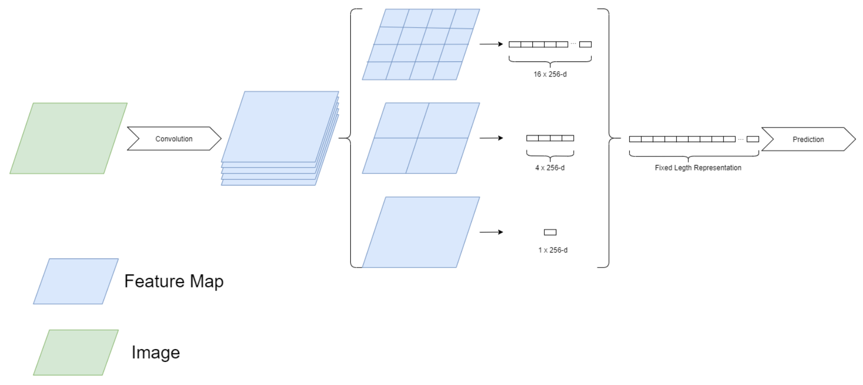

Fig. 21. SPP visualization.

One of this methods is the use of a concept known as Feature Pyramid Networks (FPN) [15]. This method aims to explore object detection on multiple feature map on different scales thus constructing a pyramid-like scheme of images or feature maps. The concept behind this is to take advantage of featurization on each level of an image pyramid. It produces a multi-scale feature representation where all levels are semantically strong. This method is proven to give accurate results while the process of each map prediction can be done in parallel so the inference time doesn't have much time penalty.

Where the FPN was a technique used on the Yolo v3 [21] with great results. The concept of Path Aggregation Network PANet [16] is present, although modified, in the neck of the YOLOV4 model replacing the FPN for this version. It's mainly incorporated in the model to enhance the process of instance segmentation by preserving spatial information. This modification suggested and implemented version of PANet is a simple modification that instead of the usual addition, it was replaced with concatenation.

The implementation of a Spatial Attention Module (SAM) serves to keep spatial attention on convolutions. The original concept was proposed on Convolutional Block Attention Module (CBAM) [27], this concept uses two modules combining the channel attention module (CAM) and then SAM. The reason behind it is that it's well known that attention plays an important role in human

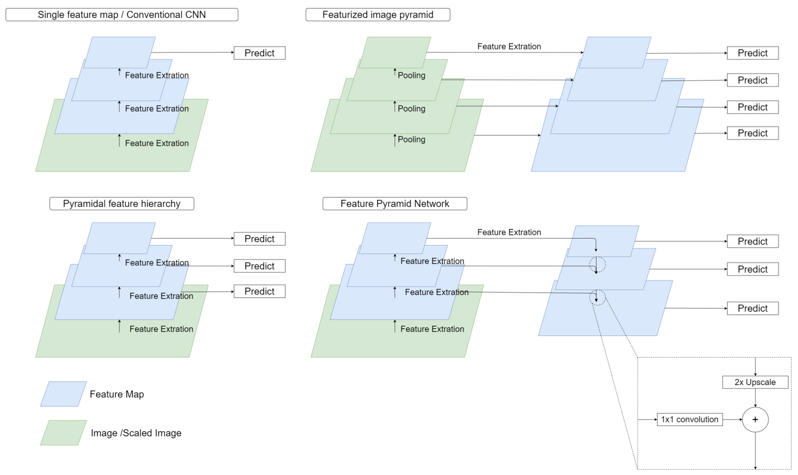

Fig. 22. CNN and Feature Pyramid Networks.

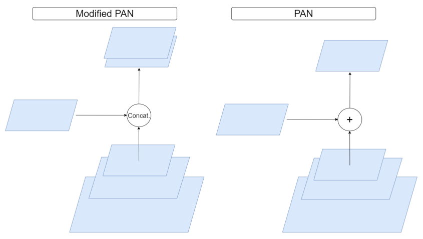

Fig. 23. modified PAN.

perception; So the development of this plugin module that can add some attention mechanisms to networks was initially the goal of the original concept. This is done by generating a single feature map by concatenating, both a max pooling and average pooling feature maps, in depth to generate a single feature map that is going to be interpreted by a convolution and normalized by sigmoid function. The authors of YOLO v4 tried to improve on this model, and were able to simplify the concept by removing the pooling operations, applying the convolutions to the input, and normalizing with a sigmoid that generates values between 0 and 1, which is later multiplied by the original input in a process similar to masking.

Now that we got the grasp of essential techniques and concepts used in this type of process, let's start with YOLO v4 where the goal of this paper was to deliver in 3 main aspects: increase the efficiency and accuracy so that could be run and trained on accessible Graphics Processing Units (GPU[2]). To achieve

[2] There are usually two types of processing unit on a Computer: the Central Processing Unit (CPU) and the Graphics Processing Unit (GPU). GPU the usual purposes is to process, as the name suggests, our graphical interfaces and render 3D applications,

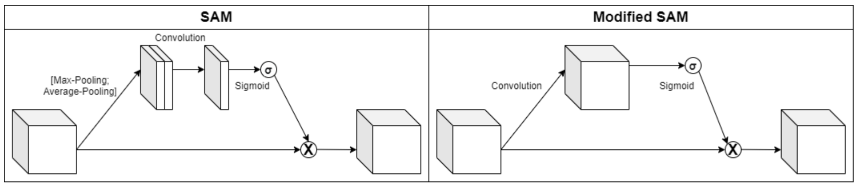

Fig. 24. modified Sam.

that, the authors modified some state of the art methods as previously explained by trying to make them more efficient and suitable for single GPU training.

YOLOV4 model consists of: CSPDarknet53 as referred for backbone; The neck which is composed with SPP, PAN; And for detection we got the YOLOv3 head.

The architectural choice made on the Darknet-53 is just a stack of convolutions, where the kernel size is either 3x3 or 1x1 followed by a residual layers.

The first step of the detector is to predict where to place the bounding box. For this to happen the network calculates the coordinates for each of the 4 corners, the feature map from the neck will be separated onto 3 parallel streams where the spatial dimension between them is different; while the detector is simply a dense network stacked on top of convolutions that reshape the feature maps. The next step is just a simple fully connected network and the model allows that each prediction box may contain multiple multi-label classifications.

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. This model purpose was to enhance the already successful Visual transformer ViT. For this field, the paper also acknowledges the success of convolution neural networks such as Alexnet but both of these solutions were for image captioning and not object detection so there was a need to create a custom solution. This model also implements some new concepts and that are just small enhancements of existent methods. Instead of global multi-head attention, the solution was to separate the image with partition windows that constitute groups of smaller patches where the size of these patches will increase on each step. So this will construct a hierarchically representation of our features like FPN. It's important to mention the windows always keep $M \times M$ patches inside and are not over-

since this are highly parallel processes, where each pixel is calculated in parallel matter while applying multiple transformations to them in real time, for that the GPU in a single die has at is disposal thousands of multiple cores (e.g Nvidia RTX 3090 as 10496 Cores). While the CPU is generally more aimed for sequentially tasks, the CPU cores can use a wider range of instructions and operations; even though the CPU cores count have been increasing along the years, the amount of cores when compared to GPU's still way under. In the consumer markets the CPU per die usually come in a 4 to 16 core configurations (e.g AMD Ryzen 5950x as 16 Cores 32 Threads).

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
1×	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
2×	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
	Convolutional	128	1 × 1	
8×	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
8×	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
4×	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. Structure of Darknet-53.(This table blatantly self-plagiarized from YOLOv3: An Incremental Improvement [21])

lapping but instead the patches themselves will increase in size.

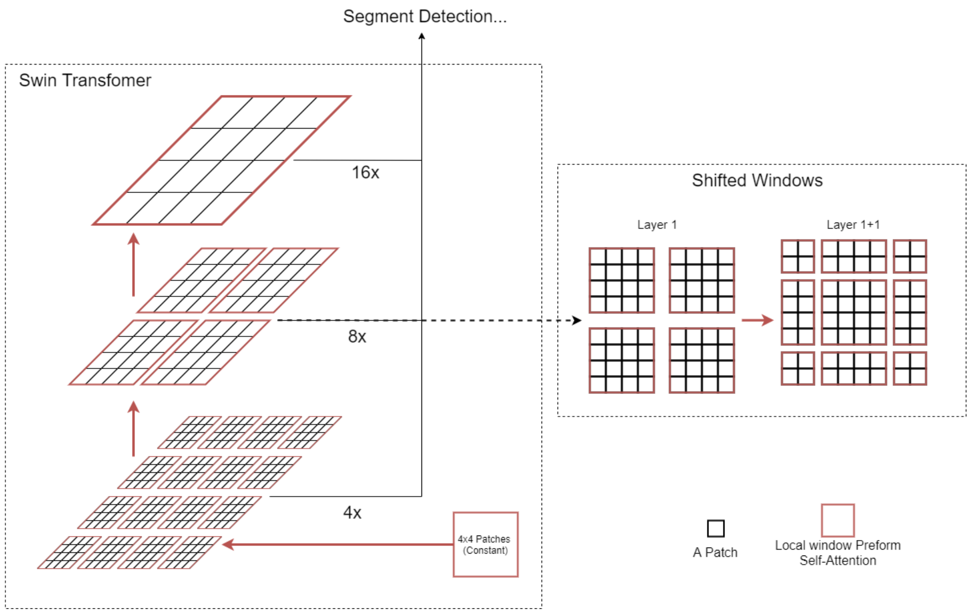

Fig. 25. Swin Hierarchical Vision Transformer using Shifted Windows

This architecture starts with a image with the size of $4 \times 4 \times 3(RGB)$, after we apply a linear transformation that the depth will be denoted as C. The Transformer blocks maintain the tokens number of $\frac{H}{4} \times \frac{W}{4}$ on each step of the process occurs a token concatenation; this process can be described as depth-wise concatenation of 2×2 neighboring blocks tokens resulting in a depth of $4C$, which is applied a linear transformation to result into a 2C. By applying this transformations it results in a down-sampling by 2x for the output dimension for the Width and height of the input.The resulting feature set has $\frac{H}{2} \times \frac{W}{2} \times 2C$. Concluding each block transforms $H \times W \times C$ to $\frac{H}{2} \times \frac{W}{2} \times 2C$ creating a pyramid of spatial representations of our input similar to FPN.

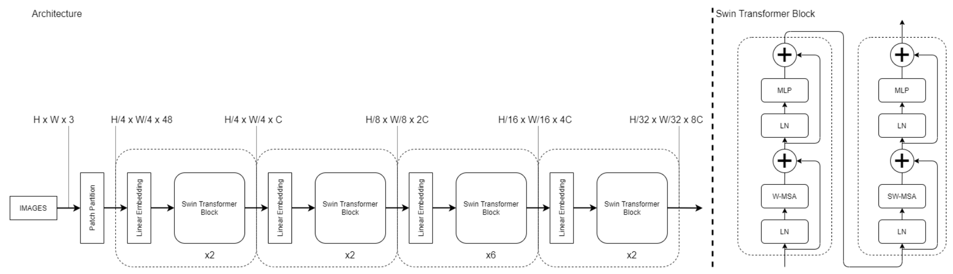

Fig. 26. Swin architectural overview

Swin Transformer block replaces the standard multi-head self attention (MSA) by a block based on shifted Window based self-attention, while the other sublayers remain the same. These small changes opposes a big problem because the model has to keep in relation the global aspect of the image, as well as the relation between each token when computing this, leading to a quadratic complexity with respect to token generation making unsuitable to some applications

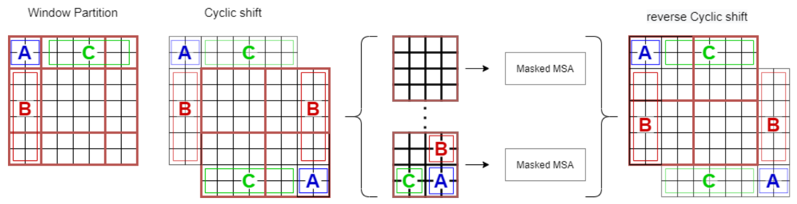

Fig. 27. Swin Shifted Windows attention

The Shifted windows work similar to the Kernel Stride, but instead the author proposed that it moves in a 2×2 patches direction (diagonally), and the image will wrap around so the top portion now appears in the bottom and the left on the right where there was empty space. This area represents where the attention mechanism will attend.

3.2 Image Captioning

We will use Image captioning model for the most of our solution this is going to be used for image-to-text transcript and to generate a description of the herbaria for cataloging reasons. Introducing the Transformer networks for image recognition, starting with the first implementations that follow the traditional Transformer, and the network the Vision Transformer (ViT) [6]. Although this network is for image captioning and not for object recognition this can be useful to do paragraph recognition. Although ViT doesn't use encoder-decoder methodologies, it just borrows the self-attention layers of the encoder. For this model to be compatible with the token like input, we need to reshape the image. This reshaping process is represented by $x \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches $x_{patches} \in \mathbb{R}^{N \times (P \times P \times C)}$ where H is height and W the weight of the original image, C is the number of channels, (P, P) is the resolution of each image patch, and $N = \frac{H \times W}{P^2}$ is the resulting number of patches, which also serves as the effective input sequence length for the Transformer. This 2D representation of our image goes through a Linear Projection to be flattened to a D size vector $X_p \in \mathbb{R}^{N \times \left(P^2 \times C\right) \times D}$.

This model also borrows the concept introduced on BERT's transformer networks [4]. The use of a learnable embedding in form of a CLASS Token embedding$z_{00} = x_{class}$, which consists on a learnable embedding, that gathers information from all the patches when attending to Multihead Self Attention (MSA) layers. The classification head simply implements a MLP with one hidden layer. The classification head only uses this hidden output from this CLASS token. The authors decided to use the relative position of the patches for the embeddings, to encode the spatial information as instead of their absolute position. They achieve this by using $1 -$ dimensional for the Relative Attention.

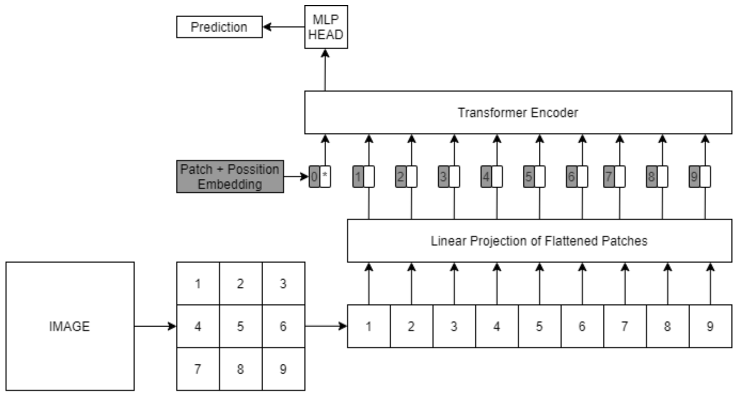
Prediction
MLP HEAD
Transformer Encoder
Patch + Possition Embedding
0 *
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
IMAGE

Fig. 28. ViT model Architecture

To improve this concept of using transformer networks for image captioning ,there were many attempts with the combined techniques of CNN + Transformer architecture like Convolutional vision Transformer (CvT) [28] and many other, we will focus in a network type that doesn't require convolutions and it was designed to do this type of operation. The CaPtion TransformeR (CPTR) [17] has the objective to replace in a certain form the traditional CNN + Transformer networks. The way this encoder-decoder transformer works is with association with words-to-patches(image patches), this means its can associate some patches of the image to the object class in words, instead of the image having just class names this model output a image caption in natural language. The architecture works by borrowing a trained weights from the ViT encoder trained previously and attaching a decoder instead of a MLP classifier the decoder is where the cross correlation between key words and image patches are going to be matched.

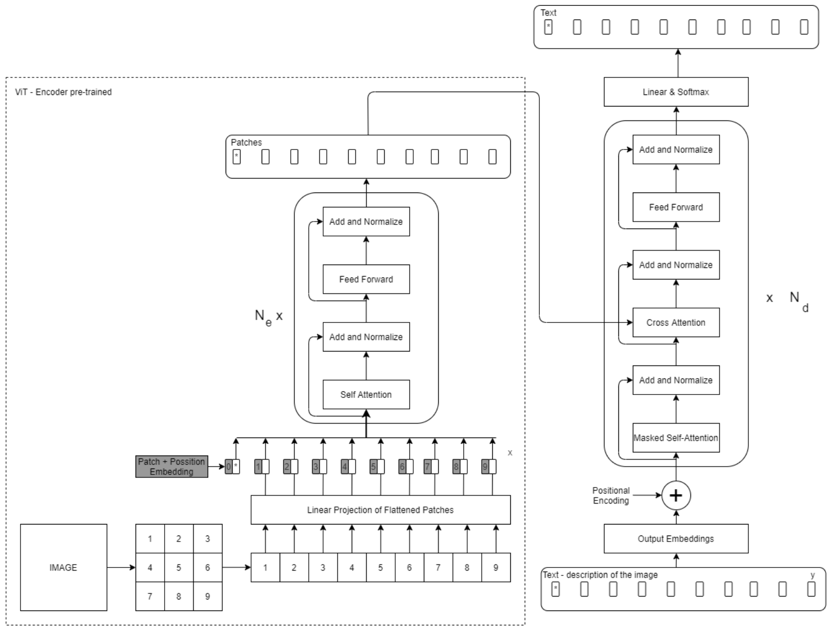

Fig. 29. CPTR model Architecture

For the decoder its similar to the original proposition. When training we encode expected results y (that corresponds to the expected image caption) with a sinusoidal positional embedding for the caption and on the cross attention is where happens the correlation with text words to the image patches, the decoder is composed with a stacked blocks with N_d times blocks. For these technique work the encoder needs to be pre-trained with the ViT model for the class identification.

I also need to feel the obligation to explain the concept used on Convolutional vision Transformer (CvT) networks since they implement a new transformer block, the Convolutional Transformer Block. This network, as proposed by the paper, doesn't require any type of positional encoding since each patch slightly overlaps each other. This process called Transformer-iN-Transformer (TNT) [7] refers that the model will need to attend for the relation of the pixels not just at a patch level, but also in a pixel level, therefore getting the relative position of each patch to patch as well as the context of the whole image.

Since this architecture works like in a sliding window token generation in a process called Tokens-to-Token (T2T) [30],by generating a single large token where we will retrieve other tokens within it, which means we can express the process from a 3d to a 2D image process with: $x_{i-1} \in \mathbb{R}^{H_{i-1} \times W_{i-1} \times C_{i-1}}$ where I mean input stage i, we learn a function $f(\cdot)$ represent the 2d convolution that maps the input $x-i$ into a $f(x-i)$ with channel size C_i the resulting width and

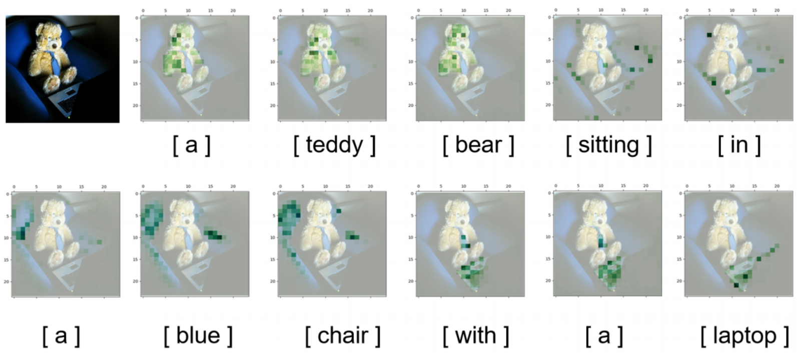

Fig. 30. Visualization of the attention blocks working and correlating parts of the image with each sentence words(This figure was blatantly self-plagiarized from CPTR: FULL TRANSFORMER NETWORK FOR IMAGE CAPTIONING [17])

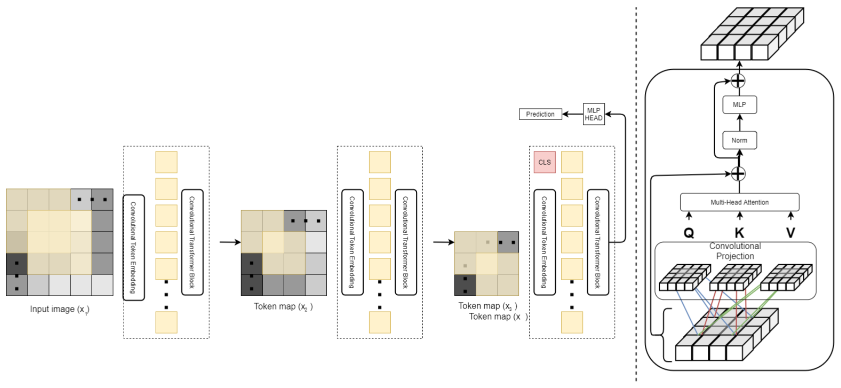

Fig. 31. CVT architecture overview & details of the Convolutional Transformer Block

height is given by a similar function of the calculation of resulting image size on the previous chapter.

$$\mathrm{H}_i = \left(\frac{\mathrm{H}_{i-1} + 2P - s}{s - o} \right) + 1 \tag{17}$$

$$\mathrm{W}_i = \left(\frac{\mathrm{W}_{i-1} + 2P - s}{s - o} \right) + 1 \tag{18}$$

Now with the 2D token map process described is followed a Convolutional Projection, using a depth-wise separable convolution layer with kernel size s, resulting into a 1D tokens.

$$\mathrm{x}^{(q/k/v)} = \mathit{Flatten}\left(\mathit{Conv2d}\left(\mathit{Reshape2d}\left(\mathrm{x_i}\right), s\right)\right) \tag{19}$$

where $x^{(q/k/v)}$ is the token input for Q/K/V matrices at layer i, xi is the resulting 2D token prior the convolution.

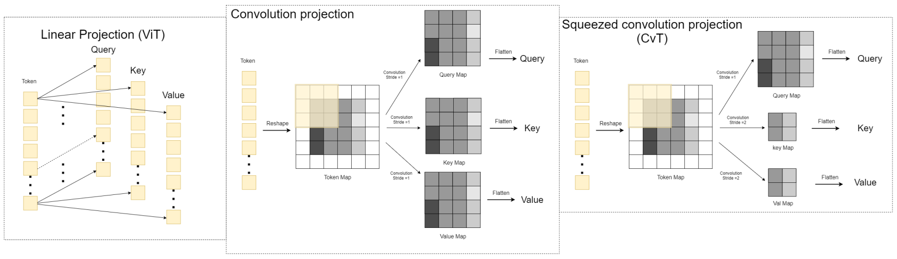

Fig. 32. Token generation CVT vs ViT

3.3 Text Recognition in Image Data

In this chapter we will describe some model that can serve to do Text Recognition in image data, we will use this models as backup due to complexity for the propose of cataloging some information that resides in region of the image that contains text (Meta-data of Herbarium Sheets). The opposing challenge is that these Meta-Data can be written by hand(Human) and/or in machine-printed letters(Machine) the other challenge we going to overcome is the language that those text can be written also taking into account some language specific characters ,just to name few challenges. We will try presenting some solutions to this model some that just apply Optical Character Recognition (OCR) with NLP in order to understand the meaning of some data for automatic label creation.

To achieve this we will explain the model proposed by Sumeet S. Singh and Sergey Karayev on Full Page Handwriting Recognition via Image to Sequence Extraction [22]. This model implements encoder-decoder transformer network. The authors decided to use a CNN encoder with a transformer decoder that is similar to the default proposition but inside there is a Localized causal self attention sub-layer instead of the normal Self Attention Layer.

The encoder since its a custom solution it works with a stack of sub-layers that follow the following arrangement; the first layers are a CNN with the use of Residual networks followed by a linear 1x1 convolution that serves just to squash our image depth wise $W \times H \times C(3D)$ to $W \times H \times 1(2D)$ after we add positional encoding to our 2D tokens to further flatten into a 1D token vector.

This model extra feature is that to the output text comes with added contextual information in relation to full page image and text, by adding markups like : <EOS >, <end-of-region>, <math>, <deleted-text>,<table>and <drawing>.<EOS >is a special markup that announces the end of the text on the page, the markups allows for easier processing.

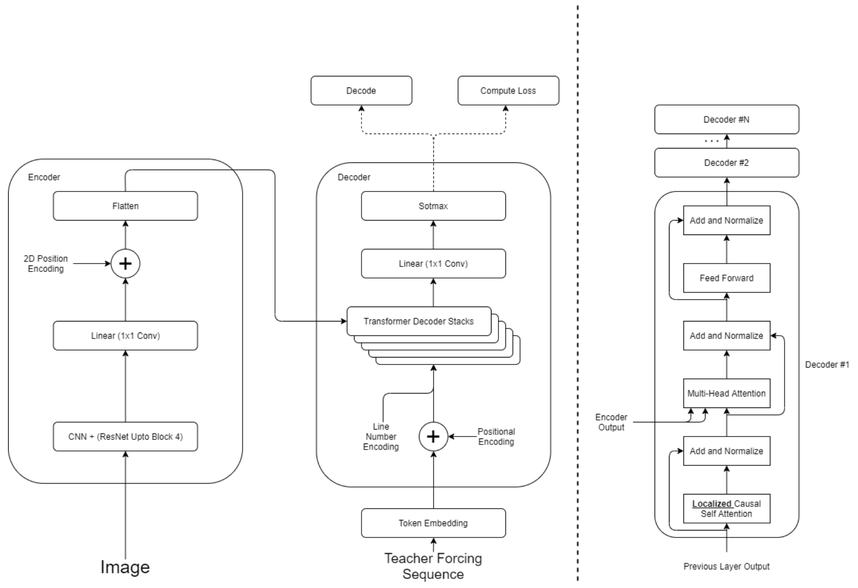

Fig. 33. Architecture Overview & Details of the Decoder Layer

Even though these model does OCR it also attends to NLP of the sentences in each step by outputting a token probability vector $\left(\vec{V}\text{- Vocabulary Set}\right)$ that represents a Vocabulary set with distribution p_t. This distribution is conditioned on past tokens $y < t$ until step I. That can be defined into:

$$\mathbf{p}_t : \{1, ..., V\} \to [0, 1]; \mathbf{Y}_t \sim \mathbf{p}_t \tag{20}$$

$$\mathbf{p}_t(\mathbf{y}_t) := \mathbb{P}(\mathbf{Y}_t = \mathbf{y}_t | \mathbf{y}_{<t}, \mathbf{a}) \tag{21}$$

$$\mathbb{P}(\mathbf{y}|\mathbf{I}) = \prod_{t=1}^{\tau} \mathbf{p}_t(\mathbf{y}_t) \tag{22}$$

This model proves to be very useful since it does computer vision for character recognition with NLP this evading the need to more complex architectures or not end-to-end models.

The next model we are going over to achieve OCR and transcript to Computational text is the model Simple Predict & Align Network (SPAN) by Denis Coquenet, Clément Chatelain, and Thierry Paquet. This model is a end-to-end non recurrence-free Fully convolutions Network (FCN), this network works on remapping a 2D(image or image patch) latent space to 1D(words) latent space.

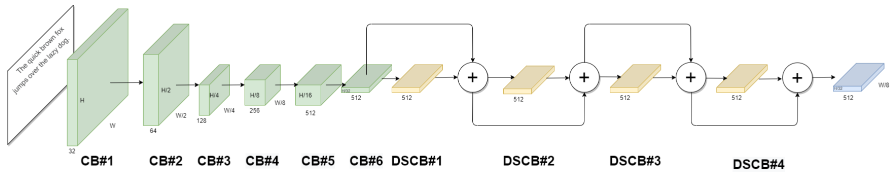

Fig. 34. Structure of FCN encoder

The SPAN encoder (not in a transformer encoder, but in a image to Tensor conversion), is composed mostly by a FCN where we do our feature extraction. This process converts our image from $x \in \mathbb{R}^{H \times W \times C}$ to $x \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{8} \times 512}$ by a succession of Convolution Blocks (CB) and Depth wise Convolution Blocks (DSCB) each of theses blocks use kernel size 3×3 with 1×1 padding that is followed by a normalization with a ReLu function. These block also dispose of with a dropout layer using Diffused Mix Dropout (DMD) [3] to prevent over-fitting. The CB as described is constituted by 3 sub-layers, one that has 2 convolution layers followed by a normalization and a third convolution layer.This layer has a stride that is different for each step of the FCN where is 1×1 for CB#1, 2×2 for CB#2 to CB#4 and 2×1 for CB#5 and CB#6. The Depth wise Convolution Blocks (DSCB) has a similar architecture as the CB but it swaps the convolution layers with Depth wise Separable Convolutions [13], this serves to reduce the number of parameters. The stride is always 1×1 independent of the block number. The output of out encoder will go though a last convolution

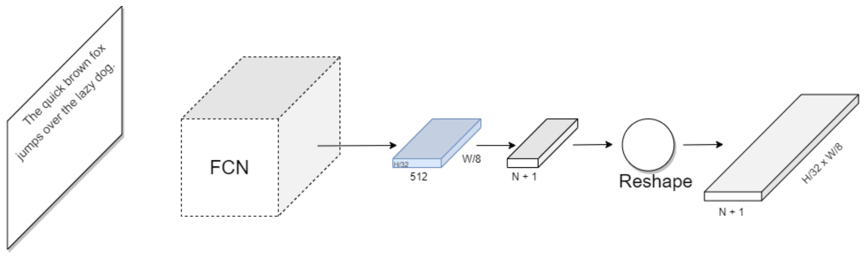

Fig. 35. Span Architecture

that will return our characters predictions. The reshape process is necessary in order to transpose the character into a length wise line like in a paragraph, the output represent a paragraph without line breaks.

3.4 Overview

Since our application is to be run on a desktop with no limitations we can choose pretty freely our model and its complexity and not be limited to Computational

Power. All the results presented will be taken from the website PaperWithCode [2] that compiles all the models from the research papers and ranks them in multiple leaderboards based on the function and the result from training using establish benchmark data sets. Now with the overview of the models that compose the SoTA on each function can be use to achieve our goal, we will go over the project architecture will be aiming to build. Due to complexity of the problem where we have multiple task to attend with different objectives our model will not be end to end. With the explanation previous presented we will need 3 or 4 models to achieve our goal. Since out project has no hard limits we will focus on the best on the following benchmarks COCO data set for object recognition where Swin is the first place on the metrics of precision with YOLOV4 by a solid 3rd place overhaul. In the field of image captioning the Vit is in a solid fourth place while the top 3 position are based on a EfficientNet we will focus on the transformer methodology to this process. For Text Recognition in Image Data we will explore some methodologies based on those presented but the difficulty to overcome will be the multiple languages.

4 Research Proposal

In this chapter, we will overview the problem and our motivation as well as present the models we aim for our solution. We'll give a small explanation of the dataset available and what can be used to train our models and the complexity of the problem. Later, we will explain how it all comes together to a model that can solve this problem.

4.1 Thesis Statement

Hundreds of herbarium collections, currently in Natural History Museums and other similar institutions, have accumulated a valuable heritage and knowledge of plants over several centuries. Recent initiatives started ambitious preservation plans to digitize this information and make it available to botanists and the general public through web portals. Such information is crucial for the study of plant diversity, ecology, evolution, and genetics. However, thousands of sheets are still unidentified at the species level while numerous sheets should be reviewed and updated following more recent taxonomic knowledge. Herbarium is a collection of preserved plant specimens and meta data used for scientific study. The digitization has a important role due to the highway of information network that the Internet can provide, our goal is to easily digitize these herbarium to be accessible to anybody and not be limited to the physical sheet in institution for research. This MSc research project will explore and evaluate the accuracy with which deep learning methods can be potentially exploited for processing herbarium images for cataloguing purposes, specifically envisioning the identification of species, and the extraction of other metadata elements for supporting the indexing of herbarium images. Particular attention will be given to the processing of labels typically present within these images, containing hand-written and/or typewritten information on the names of collectors, the date and location of the collection, and the species identification.

4.2 Datasets and Evaluation Methodology

The datasets we will use is an already established compiled collection with over 1,800 herbarium specimen [5] for the propose of cataloging. This data set has entry that date as late as 1970 and older to more recent ones, so the state of preservation of the sheets will prove a challenge, also the dataset as multiple samples of what language is written on the sheet, with 11 languages and some sheets unidentified the origins including (English, French, Latin, Estonian, German, Dutch, Portuguese, Spanish, Swedish, Russian, Finnish, Italian and a designation of ZZ for unidentified) with all this languages we also logically we have a good distribution over the globe in relation of the capture locations and species variety. See Appendixes [5] to graphical distributions of the dataset. To train our model we will use for region segmentation the 250 of the specimens that contain positional information of the labels and as the evaluation metric we will use Intersection over Union (IoU) metric that can be explained in the division of the overlapping area of the expected boundary versus the actual prediction and the union of said boundaries. To train our image captioning related to region that contain our text from meta data labels, we will use can use already human transcripted sheets. This data set is more skewed to English since its a volunteer manual work we count with more english speakers, also our metrics will be a ration of total words vs that he perfectly transcripted without errors or missing letters. At last we will have to make our own sub dataset for a image captioning of the specimen we will have to create and manual captions with the description of the specimen, we will look for details like flower, spikes and other characteristics for metrics we will count the total characteristics minus the misses divided by total.

4.3 Overview on Proposed Method

Now with the general idea of the task in hand and the complexity we will need to overcome, we will go over the idea of the solution with a brief explanation of the architecture for the model we will aim use. So we will focus on a transformers networks since the stride for this new architecture and the big success of this networks. The first process is a image region segmentation of the image in two components. A region that are composed with the specimen of the herbaria and the image region where text that the specimen and the meta data that can be hand written and/or type written from the sheet for this we will use Swin and the YOLOV4 as a back up in case we don't achieve optimal results. Secondly we will focus on region of the image where the label (image of text) where most data can be extracted for cataloging purposes. The models aimed to use is image captioning, as a back-up a more complex model like the Span and the Full Page Handwriting Recognition based on which offers best early results is what we will go with. At last, we will use a model that will caption out herbaria specimen that either be based on ViT or similiar.

5 Planning for Next Semester

st

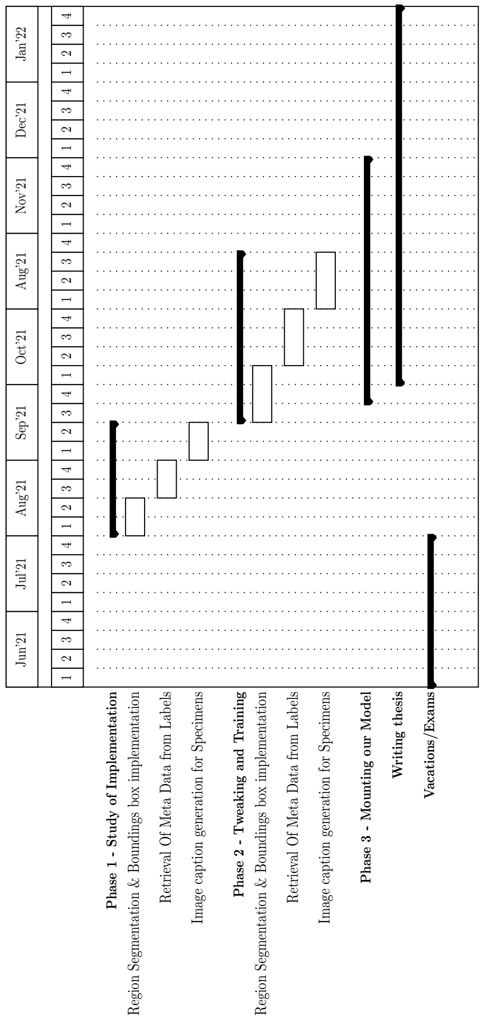

Fig. 36. Planned Schedule

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection.* 2020. arXiv: `2004.10934 [cs.CV]`.

[2] Robert Stojnic licensed under CC-BY-SA. *paperswithcode State of the art.* 2018. URL: `https://paperswithcode.com/sota`.

[3] Denis Coquenet, Clément Chatelain, and Thierry Paquet. *End-to-end Handwritten Paragraph Text Recognition Using a Vertical Attention Network.* 2020. arXiv: `2012.03868 [cs.CV]`.

[4] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2019. arXiv: `1810.04805 [cs.CL]`.

[5] Mathias Dillen et al. "A benchmark dataset of herbarium specimen images with label data". In: *Biodiversity Data Journal* 7 (2019), e31817. ISSN: 1314-2836. DOI: `10.3897/BDJ.7.e31817`. eprint: `https://doi.org/10.3897/BDJ.7.e31817`. URL: `https://doi.org/10.3897/BDJ.7.e31817`.

[6] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.* 2020. arXiv: `2010.11929 [cs.CV]`.

[7] Kai Han et al. *Transformer in Transformer.* 2021. arXiv: `2103.00112 [cs.CV]`.

[8] Kaiming He et al. *Deep Residual Learning for Image Recognition.* 2015. arXiv: `1512.03385 [cs.CV]`.

[9] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Lecture Notes in Computer Science* (2014), pp. 346–361. ISSN: 1611-3349. DOI: `10.1007/978-3-319-10578-9_23`. URL: `http://dx.doi.org/10.1007/978-3-319-10578-9_23`.

[10] Sepp Hochreiter and Jürgen Schmidhuber. "LSTM can solve hard long time lag problems". In: *Advances in neural information processing systems* (1997), pp. 473–479.

[11] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[12] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

[13] Lukasz Kaiser, Aidan N. Gomez, and Francois Chollet. *Depthwise Separable Convolutions for Neural Machine Translation.* 2017. arXiv: `1706.03059 [cs.CL]`.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[15] Tsung-Yi Lin et al. *Feature Pyramid Networks for Object Detection.* 2017. arXiv: `1612.03144 [cs.CV]`.

[16] Shu Liu et al. *Path Aggregation Network for Instance Segmentation.* 2018. arXiv: `1803.01534 [cs.CV]`.

[17] Wei Liu et al. *CPTR: Full Transformer Network for Image Captioning.* 2021. arXiv: `2101.10804 [cs.CV]`.

[18] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: `2103.14030 [cs.CV]`.

[19] David Owen et al. "Towards a scientific workflow featuring Natural Language Processing for the digitisation of natural history collections". In: *Research Ideas and Outcomes* 6 (2020), e55789.

[20] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural networks* 12.1 (1999), pp. 145–151.

[21] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: `1804.02767 [cs.CV]`.

[22] Sumeet S Singh and Sergey Karayev. "Full Page Handwriting Recognition via Image to Sequence Extraction". In: *arXiv preprint arXiv:2103.06450* (2021).

[23] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[24] Ashish Vaswani et al. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[25] Alex Wang et al. *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*. 2020. arXiv: `1905.00537 [cs.CL]`.

[26] Chien-Yao Wang et al. *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. 2019. arXiv: `1911.11929 [cs.CV]`.

[27] Sanghyun Woo et al. *CBAM: Convolutional Block Attention Module*. 2018. arXiv: `1807.06521 [cs.CV]`.

[28] Haiping Wu et al. *CvT: Introducing Convolutions to Vision Transformers*. 2021. arXiv: `2103.15808 [cs.CV]`.

[29] Fisher Yu and Vladlen Koltun. *Multi-Scale Context Aggregation by Dilated Convolutions*. 2016. arXiv: `1511.07122 [cs.CV]`.

[30] Li Yuan et al. *Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet*. 2021. arXiv: `2101.11986 [cs.CV]`.

Appendices

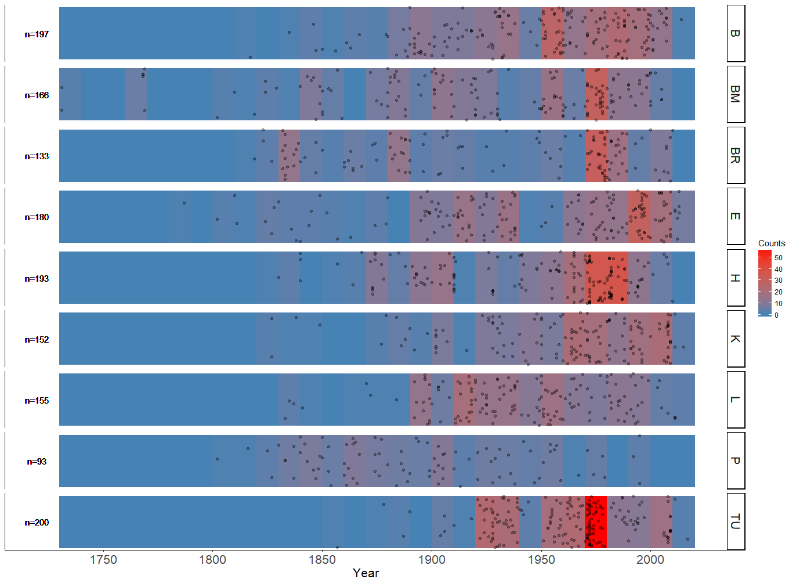

Fig. 37. Image of the distribution samples over the years with separation of language

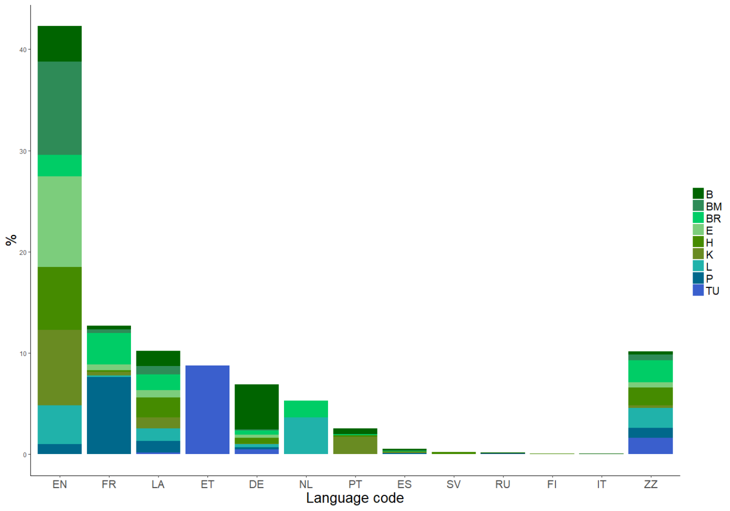

Fig. 38. Distribution of the datasets by language

This image set is self-plagiarized from A benchmark dataset of herbarium specimen images [5].

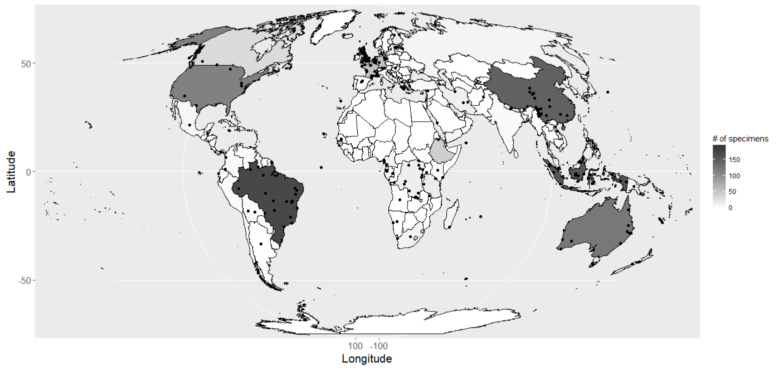

Fig. 39. Global Coverage of the specimens

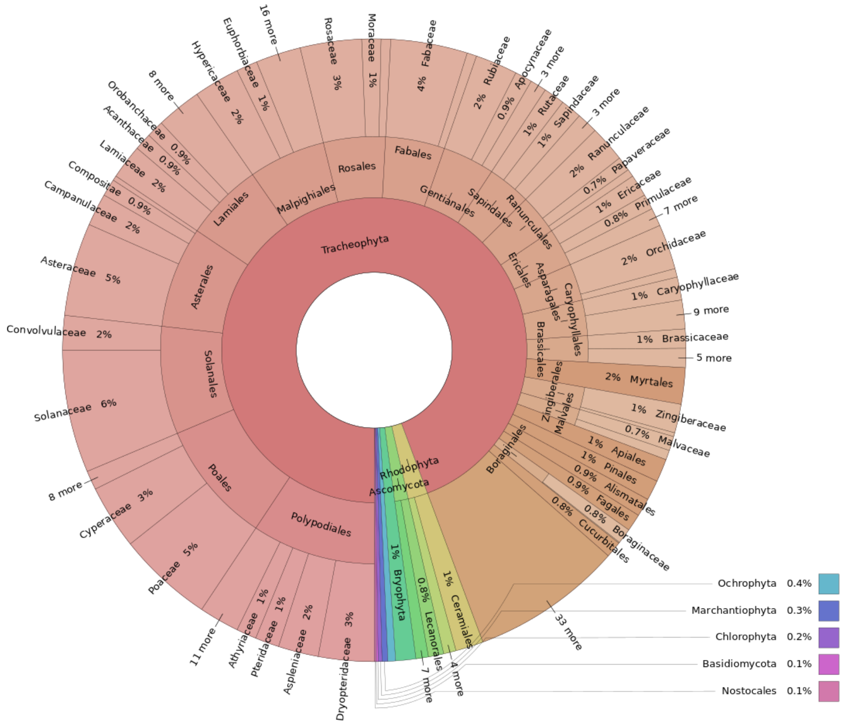

Fig. 40. Image of the distribution of herbaria specimen