S	
E1	
E2	
For Office Use Only	

Masters Project Final Report

(MCS)

2019

Project Title	Gender Classification with the First Name of the People as a Feature using Character Based 1D Convolutional Neural Network
Student Name	W.M.M.P.B Wickramasinghe
Registration No. & Index No.	2014/mcs/085, 14440859
Supervisor's Name	Dr. D. A. S. Atukorale

For Office Use ONLY

Gender Classification with the First Name of the People as a Feature using Character Based 1D Convolutional Neural Network

A dissertation submitted for the Degree of Master of Computer Science

W.M.M.P.B Wickramasinghe
University of Colombo School of Computing
2019

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: W.M.M.P.B Wickramasinghe

Registration Number: 2014/mcs/085

Index Number: 14440859

____

Signature: Date:

This is to certify that this thesis is based on the work of

Mr. W.M.M.P.B Wickramasinghe

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Dr. D. A. S. Atukorale

____

Signature: Date:

Abstract

The gender identification based on the name of a person is a traditional problem which many researchers are trying to address using different models and techniques. Convolutional Neural Network (CNN) is a powerful deep learning technique which plays a huge role in image classification and signal processing domains. The aim of this thesis is to introduce a character level one-dimensional (1D) Convolutional Neural Network model for the classification of the gender based on the name. Existing studies of the text classification using CNN are focused on sentence classification tasks with the help of two-dimensional (2D) Convolutional layers. In this study, different experiments have been administered by adjusting the model, activation functions and the dataset, in order to find an optimum model. The final model consists of 3 parallel Convolutional layers followed by max-pooling layers and 2 fully connected layers with a dropout layer in the middle. The model has been trained and tested with openly available US Census name, gender dataset. The occupied average validation accuracy is around 0.89.

Acknowledgement

I am using this opportunity to express my gratitude to everyone who supported until the end of this dissertation. Their aspiring guidance, invaluably constructive criticism, truthful and illuminating views on a number of issues was instrumental in the successful completion of this research.

First of all, I would like to give my sincere thanks to my honorific supervisor, Dr. Ajantha Athukorale who offered me advice, supervision, confidence, encouragement, guidance in right direction until the last stage of this study and in every difficulty I encountered. My thanks also goes to Dr. A.R Weerasinghe for helping me on finding a datasets for the research and his valuable feedback.

I would deeply indebted to my ever loving parents and my wife who are always there to give me courage to pursue my goals and emotional support.

I would like to thank my company ShoutOUT Labs (pvt) ltd for providing me the necessary freedom and support to complete this research.

All the members of University of Colombo School of Computing (UCSC), especially all the staff of the Post Graduate Division who helped me in many ways are gratefully acknowledged.

Finally, I am grateful to all who are always capable of giving me enough faith in doing this research at times of failure.

Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
NN	Neural Network
US	United States of America
CV	Computer Vision
NLP	Natural Language Processing
DNN	Deep Neural Network
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
NER	Named Entity Recognition
POS	Part-Of-Speech
Tanh	Hyperbolic Tangent
ReLu	Rectified Linear unit
LSTM	Long Short Term Memory
SVM	Support Vector Machine
API	Application Programming Interface
GPU	Graphics Processing Unit
RAM	Random Access Memory

I. Introduction

”Gender classification based on the name of the people” is a classical topic among the researchers. The range of gender classification research goes from designing the best features to choosing the best possible machine learning classifiers. There are 3 main types of focusing areas in these studies;

1) First name only or first name with other names based studies
2) Predicting the gender of the author by analyzing the writing style of his/her articles or posts based studies
3) Facial image based studies

The focus of this study is to create a name based gender classification model with 1D Convolutional Neural Network.

The first name of a person is a powerful indicator of his/her gender [6]. As a human, by looking at the first name of a person, we can guess the gender with little context knowledge. The first name and the last name combination can be an even good feature depending on the context whereas, in some contexts it may not. For example, in English names and Sri Lankan names, the last name is usually the family name which has no relationship with the gender of the person. However, as [15] mentioned in his study, in Indonesian names, it is not common using the family name as the last name and the last name has some sort of relationship with gender. Indonesian names also have some unisex names like ”Dwi”, ”Tri”, ”Rizki” and its variations. But there are absolutely certain gender-specific names such as, ”Putra” for males and ”Putri” for females. Therefore, in Indonesian domain, using the full name can be a good feature.

Eg: English Names
Anna Smith - Female
John Smith - Male
Eg: Sri Lankan Names
Gayan Perera - Male
Gayani Perera - Female
Eg: Indonesian Names
Hasan Suparman**putra** - Male
Sarah **Puthri** - Female

On the other hand, many researchers have found out that the Convolutional Neural Networks (CNN) are useful in extracting information from raw signals, ranging from computer vision applications to speech recognition. In modern deep learning applications, CNN plays a big role. CNN is being widely applied for the purpose of gender classification based on facial images. However, It’s really hard to find resources regarding CNN which is used for text classification related tasks. [10] studied about text classification based on CNN which show interesting results. The motivation behind this dissertation is study [10]. The model used for the research is a character based 1 dimensional CNN model.

In this research, the first name was taken into consideration and the one-dimensional CNN model was developed by considering character level features. Different parameter tuning, models and features were studied. CNN usually requires a large scale dataset for the training. It was hard to find openly available public datasets belongs to other countries such as Indonesia or Sri Lanka. Therefore, the US Census Name Gender [13] public dataset was used which has around 95,000 unique names with gender information.

The proposed model in this research created on top of the Deep Learning Studio (V 2.5.0) which is provided by Deep Cognition [16] (Section IV).

A. Problem Definition

This research defined one dimensional, character based CNN model for classifying gender labels based on the first name of the person. The first name was used as an input for the Convolutional Neural Network (CNN) and predicted the gender class, male or female. The output class had only two members; male or female. The corpus extracted from the US Census database [13] and only the unique records with non unicode characters(English alphabet characters) were taken. Size of the corpus was 95654 and it was split into three as 80% for the training, 10% for the validation and 10% for the test. A shuffle function had been used to build the data sets. The characters of the name were taken as the feature vector. The training and validation accuracy with the respective losses had been recorded.

B. Scope

Create a gender classifier model based on the first name of the people with one-dimensional Convolutional Neural Network. Different models and parameter sets were discussed and the best model and parameter set was chosen as the final model.

C. Contribution

Various deep learning models have been explored in the sentence classification domain which used Convolutional Neural Network. In the study [10] suggested a 2D Convolutional Neural Network model for sentence classification. Based on study [10], this dissertation presents a one-dimensional character based parallel Convolutional Neural Network model to classify gender based on the first name of the people. Even though, the model has been used for name, gender classification task originally, this model can be used for other word classification tasks where character based features are important.

D. Outcome

The final model was a parallel one-dimensional CNN with 3 Convolutional filters. The average validation accuracy of the model for US Census dataset [13] was around 0.89 and loss was around 0.28.

E. Thesis Organization

Section II focuses on background knowledge and structures of the CNN models and Section III focuses on previously done researches in the same domain. Section IV presents the dataset used for the research and the training environment. Section V introduces the model and the parameters used for the research and Section VI showcases the different experiments done in this research. Section VII concludes the results. Other useful resources and the source codes can be found in Section VIII.

II. Background

A. Deep Neural Network

Powerful feature learning skills have been shown in Deep Learning and remarkable performance in Computer Vision (CV) [17], speech recognition [18], and Natural Language Processing (NLP) [19] is achieved. Deep Neural Network (DNN) contains multiple hidden layers whereas shallow Artifical Neural Network (ANN) contains a single layer (Figure 1). Therefore, DNN can learn more advanced features. Some variations of DNN are Convolutional Neural Network (CNN), Recursive Neural Network and Recurrent Neural Network (RNN). Deep Neural Network consists of 2 main processes; Forward pass and Backpropagation. In the backpropagation process, the parameters of the network are updated based on the learning rate and cost function via stochastic gradient descent.

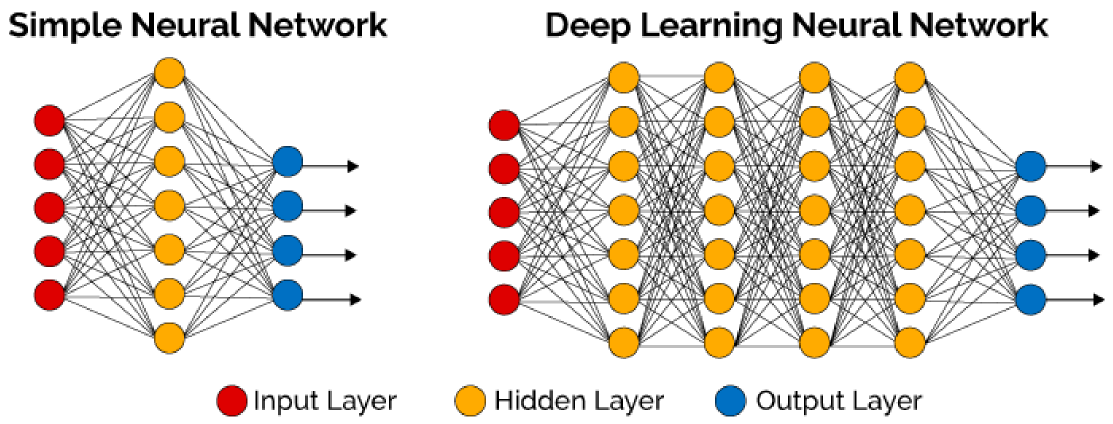


Fig. 1. Shallow Artificial Neural Network architecture vs Deep Learning Network architecture[1]

1) Convolutional Neural Network: Convolutional Neural Networks (CNNs) are a sort of feeding forward Neural Network in which each and every node can be utilized to apply filters through overlapping regions and this is generally utilizing for image classification space. The processing happens in an alternative fashion among convolution and sub-sampling layers pursued by one or more fully connected layers. This architecture has different advantages contrasted with the standard Neural Networks.

The Neural Networks (NNs) have been successfully applied to the features that have been extracted from other systems, which means that the performance of NNs depends on matching relevant features that can be obtained.

In the context of image classification, if NN is directly applied to the raw pixels, depending on the image dimension, more parameters may be required since the hidden layer is fully connected, which introduced high complexity for the model. To tackle this problem CNNs can be applied. The CNNs depend on sharing the weights, which reduces the numbers of parameters.

When classifying an image, the relationship of the nearby pixels is important. When classifying an image of a person, there are several relationships that helps to detect it as a person. For example, the mouth is below the nose and eyes are above and by the sides of the nose. These correlations can be identified by the Convolutional layers since it applies a local filter to the input image. When convolutions are applied to a certain area of the image, it will extract the local features of that area. By combining these convolved features together, it is possible to generate a less dimensional image with the same construction as the original image. These kinds of constructions are not supported by the fully connected layers.

Figure 2 represent a typical CNN architecture in the context of image classification.

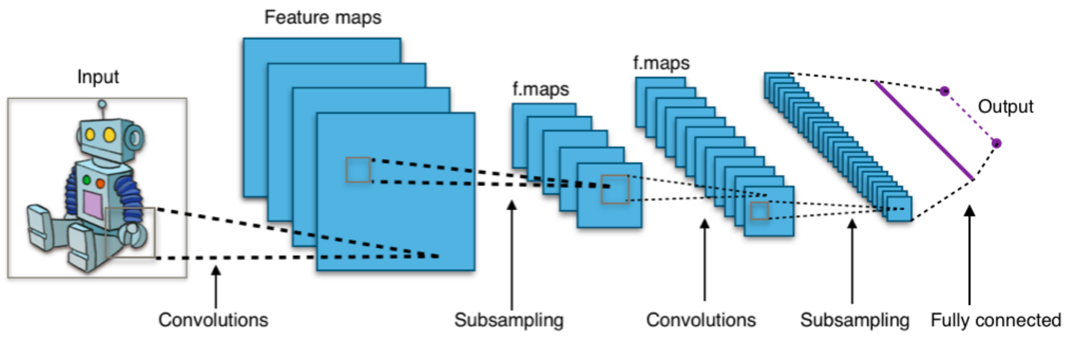


Fig. 2. Typical CNN architecture of image classification [2]

It is possible to apply the same concept from the image classification to the sentence classification with a simple 2D Convolutional Network. But for the individual text classification (character based classification), a 1D Convolutional Network which is discussed under chapter V is required.

Convolutional Neural Networks learn local features and assume that these features are not restricted by their absolute positions. In the field of Natural Language Processing (NLP), CNNs are applied in Named Entity Recognition (NER) [19], Part–Of–Speech Tagging (POS), etc.

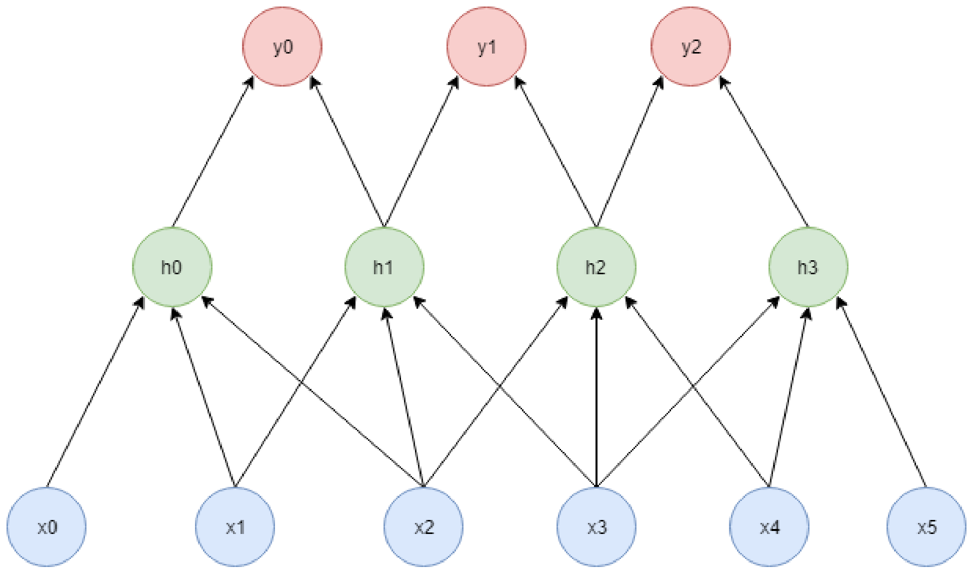


Fig. 3. 2 Layer CNN. All the neurons in each layer are locally connected with the neurons of the previous layer

Figure 3 represents a Convolutional Neural Network with 2 layers. Neurons in CNN are locally connected with neurons in the previous layer. Weights of the same filter are shared across the same layer. For any green node t, $h_t = f(W \begin{pmatrix} x_t \\ x_{t+1} \\ x_{t+2} \end{pmatrix} + b) = f(w_t x_t + w_{t+1} x_{t+1} + w_{t+2} x_{t+2} + b)$ W is shared by the same filter in same layer.

B. Architecture

The Convolutional layer, Fully Connected layer and Pooling layer are the main 3 types of components which required to build a Convolutional Neural Network. By arranging these layers in different setups and using multiple layers, different results can be taken.

1) Convolutional layer: This is the backbone of the Convolutional Neural Network. A set of learnable filters (or kernels) make the layer's parameters. These learnable filters consist of a small receptive field, which traverses through the entire input volume. There are 2 things that happen during the forward pass in each filter when it convolving across the width and height of the input volume.

- Calculate the dot product between the entries
- Generate the 2-dimensional activation map

As a consequence of this, the network learns filters which is activated by detecting a particular type of feature at some spatial position in the input.

The full output volume of the Convolutional layer is generated by stacking the activation maps for all the filters across the input volume. Thus every entry in the output volume can be presented as an output of a neuron which checkout a small region in the input and shares its parameters with other neurons in the same activation map.

In Figure 4, each blue neuron in layer m, connected with adjacent neurons which are the subset of red neuron in layer m-1. The convolution of filter and input is denoted by the connection between 2 connected neurons. Each filter has the same depth as input but smaller size along width and height.

Each filter studies a feature map. In Figure 5, weights with the similar color are shared. The gradient of a shared weight is calculated by summing up the gradients of the parameters being shared, in the back propagation process.

In Figure 6, input has size of $d_i \times w_i \times h_i = 3 \times 5 \times 5$. Input is padded with 0s of width $w_p = 1$ and height $h_p = 1$. The depth of output (or the number of feature maps to be learned) is 2. The size of filter is $d_0 \times d_f \times w_f \times h_f = 2 \times 3 \times 3 \times 3$.

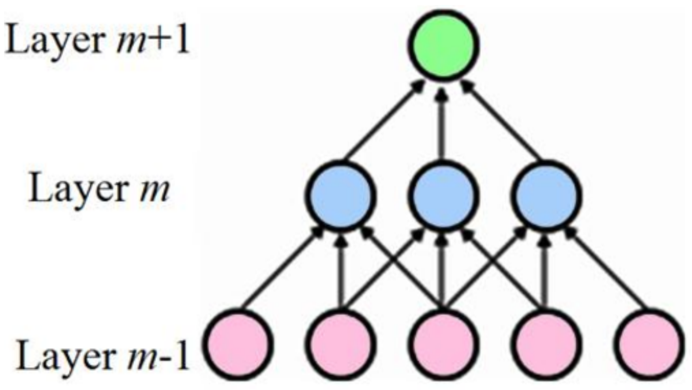


Fig. 4. Sparse connectivity in Neural Network[3]

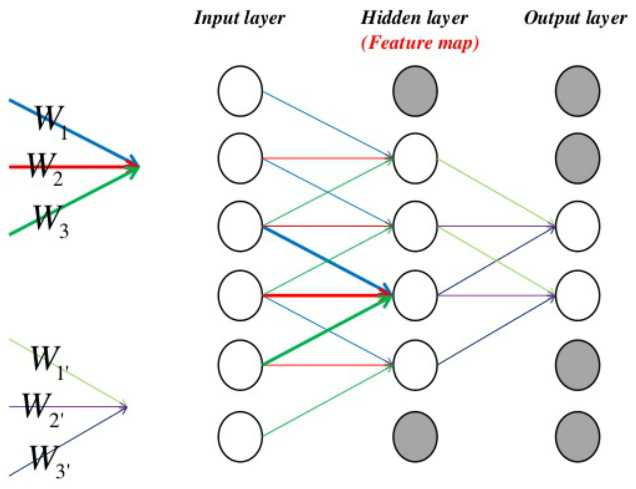


Fig. 5. Shared weight in Neural Network[4]

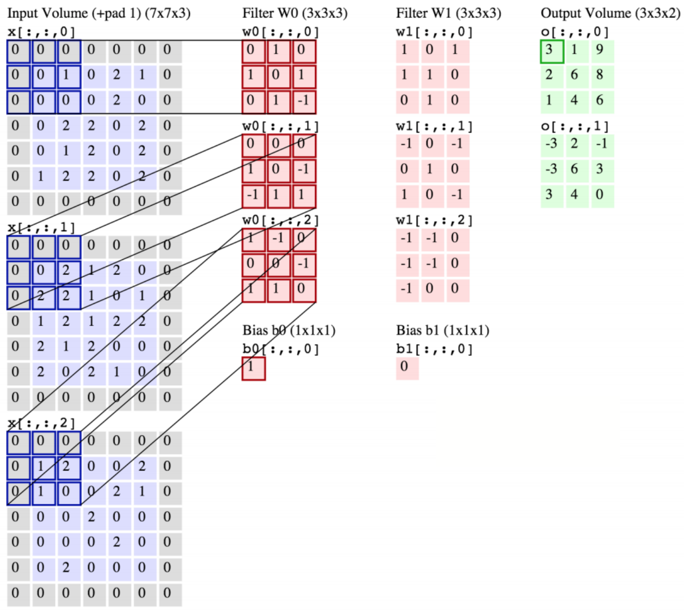


Fig. 6. Convolutional layer in Convolutional Neural Network[5]

The stride when filter is moving through the width w_s and height h_s is both equals to 1. The feature map h_k, which is learned by the filter k, is resolved by the weights W_k and bias b_k as follows:

$$h_k = f(W_k \times x + b_k)$$

, where f is an activation function which is introduced in Subsection II-C. The volume of output should be:

$$d_0 \times (\frac{w_i - w_f + 2w_p}{w_s} + 1) \times (\frac{h_i - h_f + 2h_p}{h_s} + 1)$$

The number of parameters to be learned should be:

$$(w_f.h_f.d_f + 1).o_d$$

i.e., For the output size of $2 \times 3 \times 3$, $(3 \times 3 \times 3 + 1) \times 2 = 56$ is the number of parameters to be learned.

Each and every neuron in a general Neural Network is fully connected with all the neurons in the previous layer. The number of parameters of fully connected layer become $(w_i.h_i.d_i + 1).w_0.h_0.d_0 = (7 \times 7 \times 3 + 1) \times 3 \times 3 \times 2 = 2664$, if the input and output is equal. In a general Neural Network, the number of parameters is positively correlated with the size of the input and output, whereas in the Convolutional Neural Network that is positively correlated with the size of the filter. When compared to the size of the input and output, the size of the filter is small. Each and every layer is fully connected with the adjacent layers in a general Neural Network making the number of parameters in it very large. This sometimes cause the learning process over-fitting.

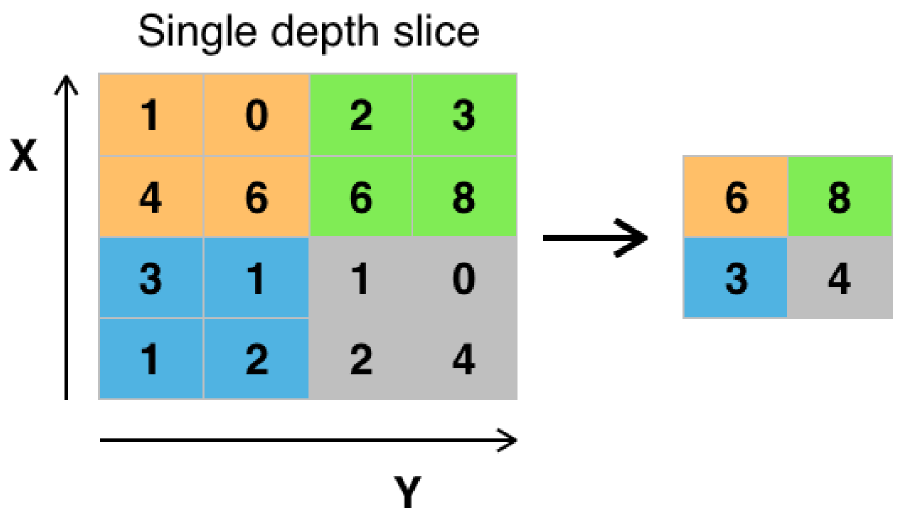


Fig. 7. Example max pooling with 2x2 filters and stride=2 [2]

2) Pooling layer: Convolutional Neural Networks may consist of local or global pooling layers. A single neuron in the next layer is generated by combining the output of each neuron cluster in the layer. For example (Figure 7),

- Max pooling - maximum value from each cluster at the layer is used to generate the next layer
- Average pooling - average value from each cluster at the layer is used to generate the next layer

Pooling is a method of non-linear down-sampling. Several non-linear functions are used for the implementation of pooling. The max pooling is the most common pooling method. It divides the input image into a set of rectangles which do not overlap each other. The maximum value of each set of the rectangle will be the output.

The rough location of the feature relative to the other features is critical than the exact location of the feature. This is the reason behind the utilization of pooling in Convolutional Neural Networks. This layer decreases,

- Spatial size of the representation
- Number of parameters
- Amount of memory and computation in the network

This results in over-fitting control. In Convolutional Neural Network architecture, it is usual to add a pooling layer between Convolutional layers. The pooling process gives another form of translation invariance.

The pooling layer with filters of size 2×2 is the most common form (Figure 7) which is applied with the stride of 2 down samples throughout the input width and height. This results in discarding 75% of the activation. In this example, the dimension of the depth is not changed and all max operations run covering 4 numbers.

The pooling units can utilize other functions in expansion to max pooling, such as $\ell 2$-norm pooling or normal pooling. Normal pooling was frequently utilized historically. But max pooling is being used widely since it performs well in practise.

The height, depth and width of input are h_i, d_i and w_i respectively. The pooling size is $d_p \times w_p \times h_p$. Usually $d_p = d_i$. The stride of pooling has the size of $w_s \times h_s$. Zero–padding for Pooling layers are not common in use.

The output should have size:

$$d_i \times (\frac{w_i - w_p}{w_s} + 1) \times (\frac{h_i - h_p}{h_s} + 1)$$

In most cases, input is pooled non–overlapping, i.e., $w_s = w_p$ and $h_s = h_p$. So the output has size of:

$$d_i \times \frac{w_i}{w_p} \times \frac{h_i}{h_p}$$

, which reduces the size of output by $\frac{1}{w_p . h_p}$.

3) Fully connected layer: Convolutional Neural Network always consists of several fully connected layers following each Convolutional layer. The neurons of the fully connected layers are completely joined with the neurons in the previous layer. The structure of the fully connected layer is as same as a layer in a regular Neural Network.

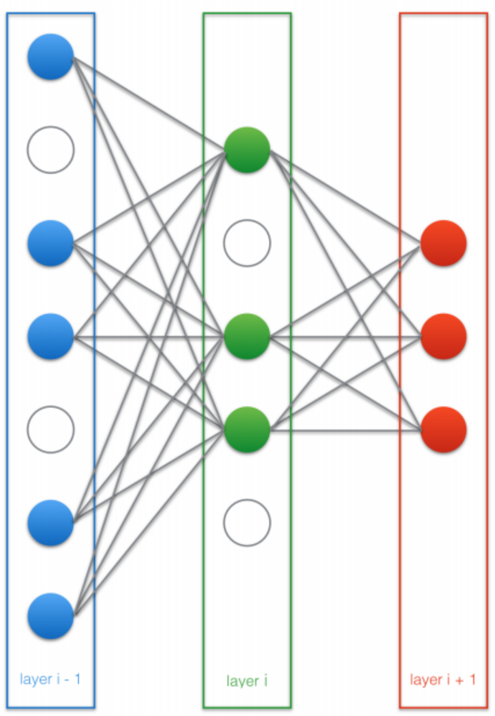


Fig. 8. Dropout process. Dropout has been applied to layer i-1 and layer i [5]

4) Dropout layer: Dropout (Figure 8) is a regularization technique which is used to avoid model over-fitting. In the training process, when applying a dropout, it will temporarily ignore (dropped out) few randomly selected neurons according to the specified probability P in the layer. The contribution of these selected neurons will not be available to the downstream neurons during the forward pass and weight update will not be applied to the neurons in the backpropagation. ”Temporarily” means, the selected unit is only dropout when training that particular sample.

A Fully connected layer possesses the vast majority of the parameters, and consequently, neurons create codependency among one another during the training which controls the individual intensity of every neuron prompting over-fitting of training data.

In general, Dropout layer is applied only for the fully connected layers and is not applied for the Convolutional layers, Pooling layers or the last Fully Connected layer.

C. Activation functions and cost functions

Activation functions are required to introduce non-linearity to the network. Without a non-linearity in the network, even a multi-layer Neural Network will perform as a single layer network. Activation functions convert an input signal into an output signal (Figure 9).

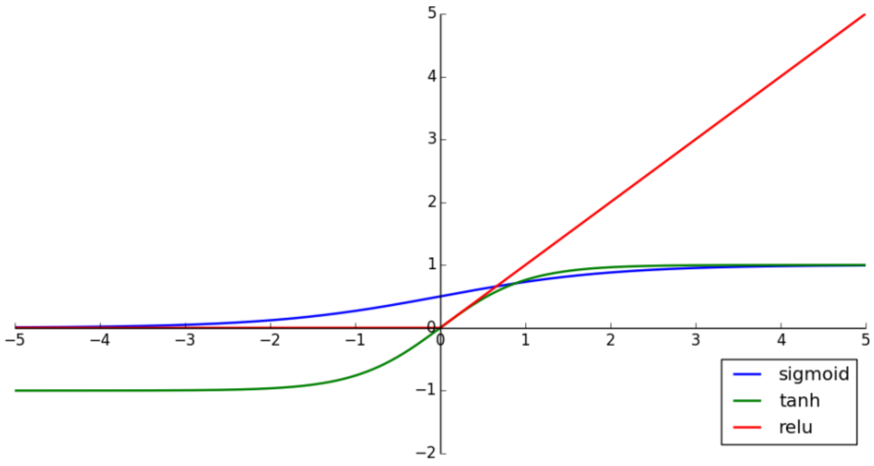


Fig. 9. Activation function graphs for Sigmoid, Tanh and ReLU [5]

Following functions have been evaluated in this study.

1) Sigmoid function: The value of the Sigmoid function varies between 0 and 1 which is ideal for predicting probability, since probability differs between 0 and 1.

$$f(z) = 1/(1 + exp(z))$$

$$f'(z) = f(z)(1 - f(z))$$

$$f : \Re \rightarrow [0, 1]$$

2) Hyperbolic Tangent functions (tanh): Tanh also a sigmoidal shape function in which value changes between -1 and 1. This is useful for the binomial class (2 class) problems.

$$f(z) = tanh(z) = (e^z - e^{-z})/(e^z + e^{-z})$$

$$f'(z) = 1 - f(z)^2$$

$$f : \Re \rightarrow [-1, 1]$$

3) Rectified Linear unit (ReLu) function: ReLu is broadly using in the Convolutional Neural Network and Deep Learning domain. This is a half rectified function. The range of the Relu can differ between 0 and infinity. The values are which less than zero becomes zero.

$$f(z) = max(0, z)$$

$$f'(z) = \begin{Bmatrix} 0 & \text{z} < 0 \\ 1 & \text{z} \geq 0 \end{Bmatrix}$$

4) Softmax function: Useful for the multi-class problems. Softmax will calculate the probabilities of all possible target classes. Function range is between 0 and 1.

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$
j=1,2,3,....,K

5) Binary Cross Entropy - Cost function:
The Binary Cross Entropy loss function (cost function) is more popular for the multi-label tasks. In this study, Binary Cross Entropy has been used as the loss function.

$$z(t, o) = -(tlog(o) + (1-t)log(1-o))$$

D. Validation

1) K-Fold cross validation: Mainly the dataset split into 2 parts as training and validation. Training dataset is used for the training of the network and validation set is used for checking the performance of the trained model. If these 2 datasets mixed with each other, then the results can be extremely biased. Therefore, these 2 datasets should be kept separately. However, there's a drawback in this case as validation data will never be used for the training of the network. Data is expensive. Hence not utilizing even one dataset which can be used for optimization of the network is a waste. The solution for this is using K-fold cross-validation. In this approach, the dataset will be split into several parts and all the data partitions will be used for training and validation of the network. The average value will be the final result. K-Fold cross-validation is very useful when having a limited dataset to train the model.

The Figure 10 shows the representation of a 5-Fold cross validation.

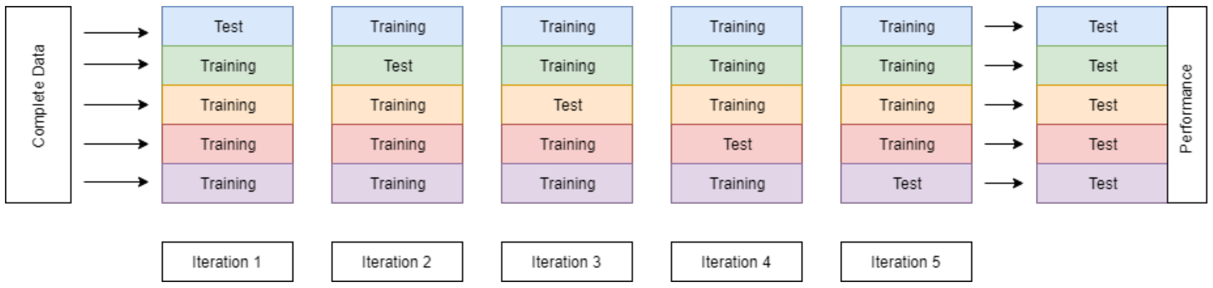


Fig. 10. 5-Fold cross validation process. Selection of training and test data sets in each iteration

In K-Fold cross-validation, the K represents the number of folds and it is an unfixed number. K can be changed according to the characteristics of the specific dataset. The most common value of K is 10. In this technique, the full data corpus is divided into K partitions (folds) in "stratified" manner with labeling each partition. The dataset in each partition and the label which is assigned to fold is kept as same throughout the training. There will be K iterations in the training process. In each iteration, the data in one fold which is not chosen as a test fold previously will be taken as the test dataset and data in all the other folds (construction set) will be taken as the training dataset. After the K training iterations, the average of all the results will be taken as the final result.

E. Common CNN architectures

Figure 11 shows the most common Convolutional Neural Network architecture pattern.

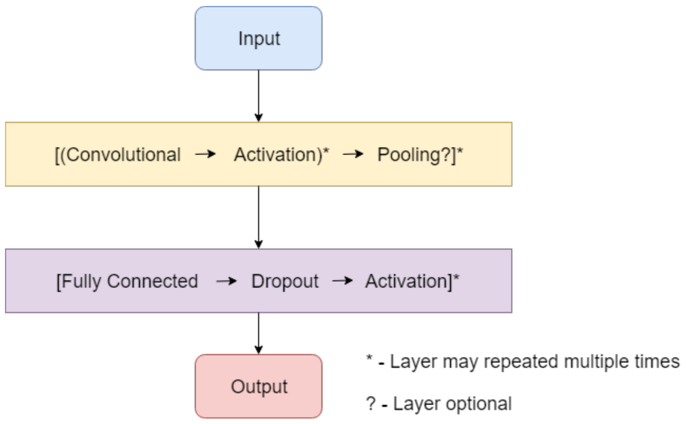

Input
[(Convolutional → Activation)* → Pooling?]*
[Fully Connected → Dropout → Activation]*
Output
* - Layer may repeated multiple times
? - Layer optional

Fig. 11. Common Convolutional Neural Network arhitecture

According to Figure 11, there will be one or more Convolutional layers following with or without a Pooling layer after the Input. Then one or several fully connected layers with or without Dropout layer and finally the Output layer.

III. Related Works

In a study by [14], 3 classifiers were compared (Naive Bayes, Maximum entropy and Decision tree) for gender classification based on the name and resulted accuracy was about 75%- 80% (Table I). They had used a dataset of 8000 records. As of their results, maximum entropy performed well compared to the other two algorithms.

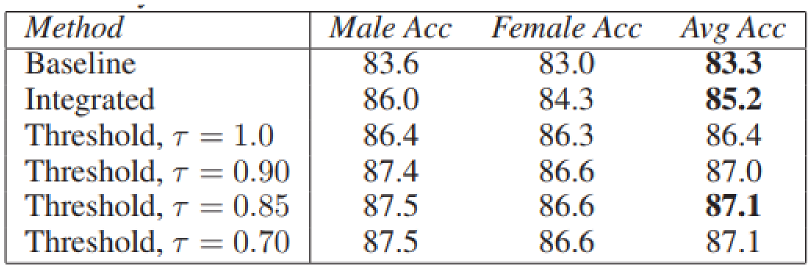
TABLE I
Resulted accuracy of Naive Bayes, Maximum Entropy and Decision Tree Classifiers in the study of [14]

Training Set	Naive Bayes Classifier	Maximum Entropy Classifier	Decision Tree Classifier
501 - 750	0.7555	0.7956	0.7856
501 - 1000	0.7595	0.7675	0.7595
501 - 1300	0.7695	0.7675	0.7675
501 - 1500	0.7515	0.7515	0.7515
501 - 1800	0.7735	0.7795	0.7775
501 - 2000	0.7896	0.7996	0.7996

[15] studied a model based on character-level Long- Short Term Memory (char-LSTM). The dataset he had used was an Indonesian name with gender and the dataset size was around 7000 records. In this study, he mentioned a few specific characteristics in Indonesian names that differ from English names. In Indonesian names, it is not common to use family names as the surname. Therefore he compared two approaches, using only the first name and full names. Further, Indonesian names also have unisex name variations as well as full confident name variations. Instead of 2-5 character n-gram features which are used commonly in relevant studies, they had used first and the last letter of first and last names as basic features. As the replacement for 2-5 n-grams, they had selected 1000 top features with the highest values of chi-squared statistics test. With the full name approach, they achieved the accuracy of 92% and for the first name approach, the resulted accuracy was about 90%.

Naive Bayes and Random Forest classifiers were compared in a study by [20] using a dataset of Indonesian names with gender and achieved 70%, 83% of accuracy respectively. In this research, he had studied the letter occurrence frequency in the names and found out that letters ’a’, ’u’, ’e’, ’t’, ’l’, ’i’ occurred more often in female names than in males and letter ’b’,’c’,’o’ appeared more frequently in male. Also, the last letter was distinctive between males and females. Therefore, the number of occurrence of characters ’a’, ’u’, ’e’, ’t’, ’l’, ‘b’, ‘d’, ‘o’ and the index of the alphabet of the last letter selected as the features. Other additional features are the second last letter and the first 2 letters. The dataset has 50,000 Indonesian name records.

[6] built a gender-name association score from first names in their study and they proposed three classifiers named as Baseline, Integrated and Threshold which were based on Support Vector Machine (SVM). In the Baseline approach they ignored name information and in Integrated approach, the gender-association score of the name was added to the user’s feature vector as a separate element. In the Threshold approach, the gender-association score of the user’s name was used to decide whether the usage of SVM-based classifier was needed. As of the results in Figure 12, the maximum average accuracy was given by the Threshold method.



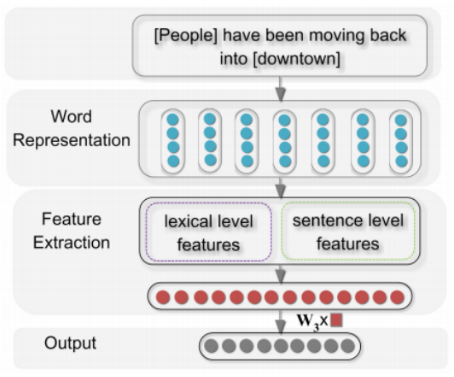
Method	*Male Acc*	*Female Acc*	*Avg Acc*
Baseline	83.6	83.0	**83.3**
Integrated	86.0	84.3	**85.2**
Threshold, $\tau = 1.0$	86.4	86.3	86.4
Threshold, $\tau = 0.90$	87.4	86.6	87.0
Threshold, $\tau = 0.85$	87.5	86.6	**87.1**
Threshold, $\tau = 0.70$	87.5	86.6	87.1

Fig. 12. Performance comparison of 3 methods (Baseline, Integrated and Threshold) in the study [6]

Many authors had investigated gender classification by using text sentiment in blogs, articles, and forum platforms [21], [22], [23], [24], [25], [26], [27], [28], [29]. Those authors explored a variety of methods, including word frequencies, writing styles, Part-Of-Speech (POS), n-gram, POS tags, unigrams, word frequencies, word classes, POS patterns, POS contents and POS style metrics to analyze text.

Convolutional Neural Network is quite popular in image classification domain. However, some studies had been carried out where CNN applied in POS tagging [30], chunking, Named Entity Recognition (NER), semantic role labeling [19], searching queries and Web documents [8], sentence classification [31][10], semantic modelling [12], relation classification [32], and other NLP tasks.

A. Single Convolutional layer CNNs



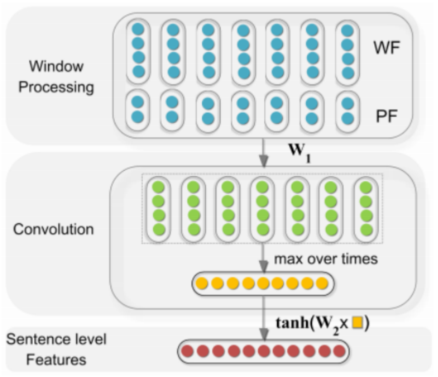


Fig. 13. Neural Network for Relation Classification (left) and Framework for Extracting Sentence Level Features (right) [7]. In the right hand figure, WF stands for word features and PF stands for position features

Neural Network for question classification had been studied by [7]. The Relational classifier took a sentence as an input and discovered multiple levels of feature extractions where higher levels represent more abstract aspects of the inputs. Lexical level features were extracted by using word embedding and the sentence level features were learned by a max-pooled Convolutional Neural Network. Figure 13 shows the overall architecture. Each token represented as Position Features (PF) and Word Features (WF) in the Window Processing Component. Then, the vector went through a Convolutional component and the sentence level features were obtained through a non-linear transformation. The final step was sending sentence level features and lexical level features into the Neural Network. The output was the prediction of the connection between 2 given nouns in the sentence.

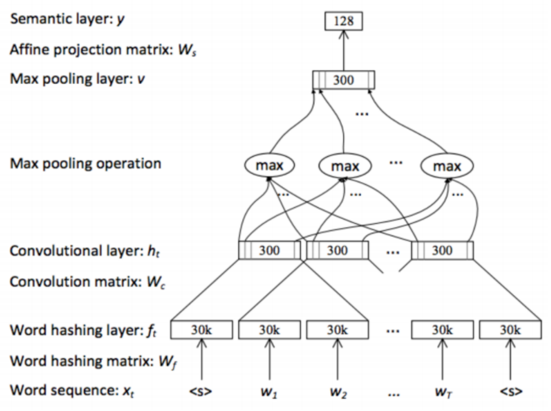

Semantic layer: y
Affine projection matrix: W_s
Max pooling layer: v
Max pooling operation
Convolutional layer: h_t
Convolution matrix: W_c
Word hashing layer: f_t
Word hashing matrix: W_f
Word sequence: x_t
128
300
max
30k
<s>
w_1
w_2
w_T

Fig. 14. CNN model in study [8] for web document searching

[8] and [9] presented similar Convolutional Neural Networks. Both of them transformed the word into a vector using letter–trigram. Then the word vectors were fed into a 3 layer Convolutional Neural Network (Convolutional layer $\rightarrow$ Max Pooling layer $\rightarrow$ Fully Connected layer as the output layer). Figure 14 shows the model for queries and web documents searching [8]. [9] tested the model on a question set from a commercial search engine. [9] used a question dataset and trained a model for relation extraction and another model for entity extraction, as shown in Figure 15. The authors defined this problem as a multi-class classification, i.e., given a query returning one relation each time while returning 150 top-scoring candidates.

[10] trained a network with one convolutional layer followed by a max–over time pooling, a fully connected layer with dropout and softmax output layer for sentence classification, as shown in Figure 16. This “one convolutional layer” consisted of 3 parallel convolutional layers with different filter sizes. The model was trained with two channels and only the parameters of one channel were updated in the training progress. The word2vec was input feature.

B. Multi convolutional layer CNNs

A sentence matching model with Convolutional Neural Network had been studied by [11]. This model contained 1 dimensional Convolutional layer $\rightarrow$ 1 dimensional Max Pooling layer $\rightarrow$ multiple 2 dimensional Convolutional layers $\rightarrow$ Pooling

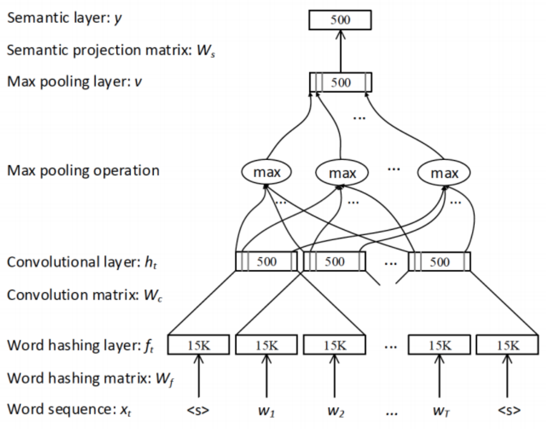

Semantic layer: y
Semantic projection matrix: W_s
Max pooling layer: v
Max pooling operation
Convolutional layer: h_t
Convolution matrix: W_c
Word hashing layer: f_t
Word hashing matrix: W_f
Word sequence: x_t
500
max
15K
<s>
w_1
w_2
w_T

Fig. 15. CNN model in study [9] for question classification

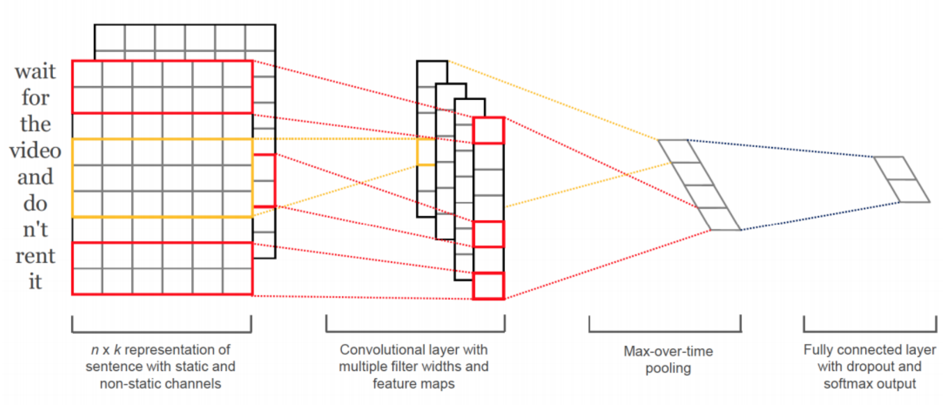

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Fig. 16. CNN model for sentence classification [10]

layer → multiple Fully Connected layers as shown in Figure 17. The embedding of words in the sentences was the input and the matching degree was the output of the network. The approach was tested on sentence completion [33], matching a response to Weibo, and MSRP dataset [34].

[12] designed a Dynamic k–Max Pooling Convolutional Neural Network (DCNN) for sentence modeling. The authors used multiple one–dimensional Convolution layer → feature maps folding operation → k–max pooling layer → a fully connected layer as output, which is shown in Figure 18. The k highest values from the inputs were chosen using K-max pooling and their original orders were kept as same. Twitter sentiment prediction task (negative and positive labeled tweets) and SST–1, SST–2, 6–type question categorization in the TREC dataset was used to evaluate this model. Dynamic k–Max Pooling Convolutional Neural Network's (DCNN) performance was good at TREC and SST-1 whereas not well at SST-2 dataset when compared with model of [10]. Convolutional Neural Network with a single convolutional layer showed good performance on various tasks. The design of [10] made the difference from others since it had parallel convolutional layers.

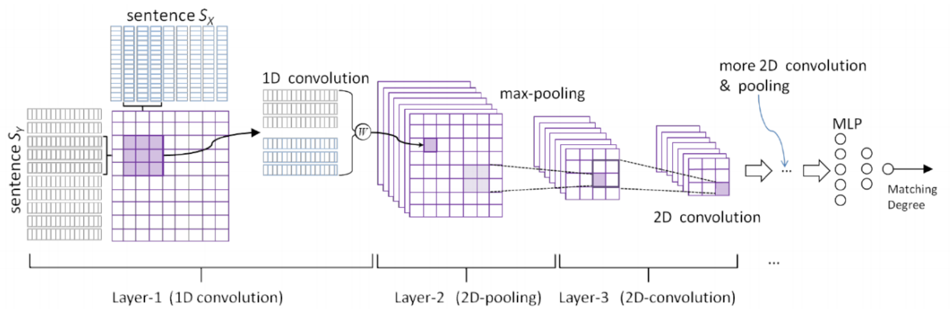


Fig. 17. ARC II model in study [11] for sentence matching

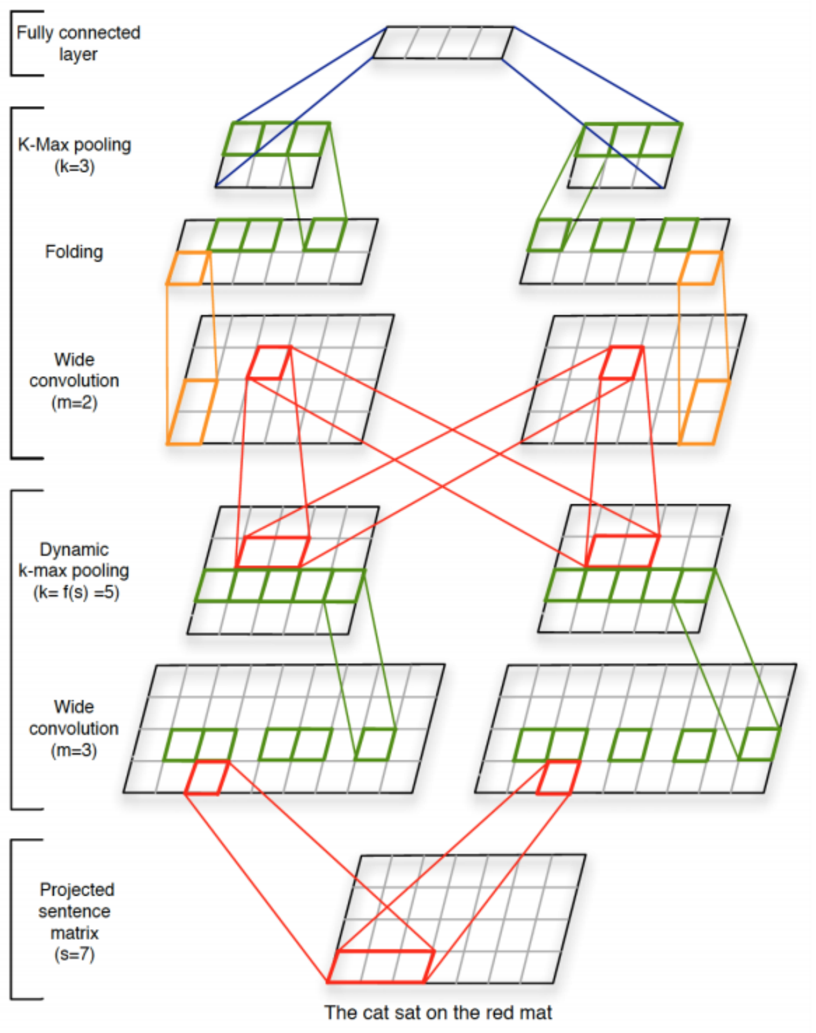

Fully connected layer
K-Max pooling (k=3)
Folding
Wide convolution (m=2)
Dynamic k-max pooling (k= f(s) =5)
Wide convolution (m=3)
Projected sentence matrix (s=7)
The cat sat on the red mat

Fig. 18. Dynamic k–Max Pooling Convolutional Neural Network model in the study [12] for modeling sentences

IV. Dataset and Environment

A. Dataset

The focus of this research was to classify a given name into male or female. Therefore, name and gender labeled dataset was required. The openly available US Census name gender [13] had been used.

1) US Census Name Gender Dataset: This dataset was provided by the United States Census Bureau. These data were extracted from the Social Security card application for births in the United States from 1879 to the end of February 2016. The total size of the unprocessed dataset was around 26 million records.

US Census database had the following data format.

First Name , Gender , Frequency

Eg: Madura, Male, 8549

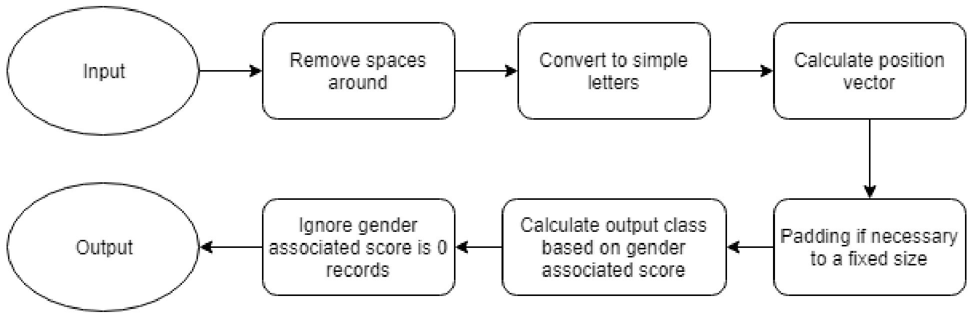

Input
Remove spaces around
Convert to simple letters
Calculate position vector
Padding if necessary to a fixed size
Calculate output class based on gender associated score
Ignore gender associated score is 0 records
Output

Fig. 19. Data preparation flowchart for the model

2) Data Preparation: Each and every record in the corpus ran through a preprocessing (Figure 19) which cleaned up unnecessary white spaces, special characters and converted into lower case letters. Two class names (Male, Female) were converted into class index 1 and 0 respectively.

Then, all the records in the corpus were converted into the input vector of fixed length with zero padding (if necessary). Each character of the input vector was replaced by the position index of the character in the English alphabet and separated each character by a semicolon (;). The class index was attached to the vector and separated by a comma (,).

Eg: with fixed length 9, the male name ”John” was converted into an index vector as follows.

10;15;8;14;0;0;0;0;0,1

English alphabet position vector is shown in Table II.

When converting the US Census database into input vectors for training the model, the gender score model had been used in [6]. By using the gender score, it was possible to classify the duplicate names in both classes based on the maximum frequency class. The names which had similar frequency had been ignored. [6] gender score model works as below.

The gender association of name x is given by the formula,

$$(M(x) - F(x))/(M(x) + F(x))$$

$M(x)$: Number of time a name given to the male $F(x)$: Number of time a name given to the female

The score ranges from -1 (the name was only given to females) to 1 (the name was only given to males). One useful property of this definition of gender-name association score was that it was clear the way to assign a score to a name for which there were no observations at all. If the score was 0, there was a likelihood that the name belonged to a male or female.

The gender score distribution of the dataset shown in Figure 20.

B. Tools and Environment

1) Deep Learning Studio: The model was created using the Deep Learning Studio (V 2.5.0) which was provided by Deep Cognition [16]. This was a freely available tool to install and able to run on own machine. It was just a User Interface wrapper for the well known deep learning tool Keras [35]. Keras was a high-level Neural Network Application Programming Interface (API), written in Python and capable of running with TensorFlow, CNTK, or Theano. Deep Learning Studio was a self-contained package which could be automatically installed all its dependencies in a Virtual Machine environment.

2) CUDA: In order to run experiments on Graphic Processing Unit (GPU), CUDA driver and CUDA Toolkit were needed for Nvidia’s GPU–programming toolchain. CUDA Toolkit was downloaded from developer.nvidia.com, which contained an nvcc program – a compiler for GPU code.

TABLE II
ENGLISH ALPHABETIC CHARACTERS WITH ITS POSITION INDEX WHICH WERE USED IN THIS STUDY

Character	**Position Index**
a	1
b	2
c	3
d	4
e	5
f	6
g	7
h	8
i	9
j	10
k	11
l	12
m	13
n	14
o	15
p	16
q	17
r	18
s	19
t	20
u	21
v	22
w	23
x	24
y	25
z	26

3) Experiment Environment: The specification of the computer which was used to ran the tests as follows.

- Operating System : Windows 10 64 bit
- RAM: 8GB DDR4
- Processor: Intel(R) Core(TM) i7-8550U CPU @ 1.8 GHz 1.99 GHz
- GPU: NVIDIA GeForce MX 150 2GB

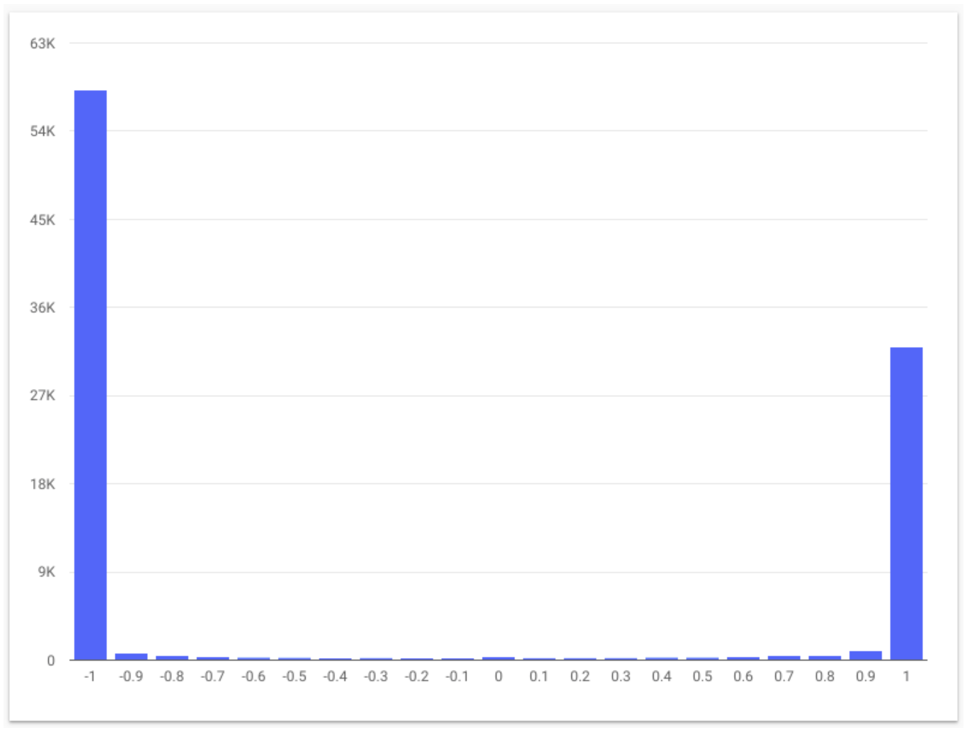


Fig. 20. Gender score distribution of US Census dataset [13]

V. Model

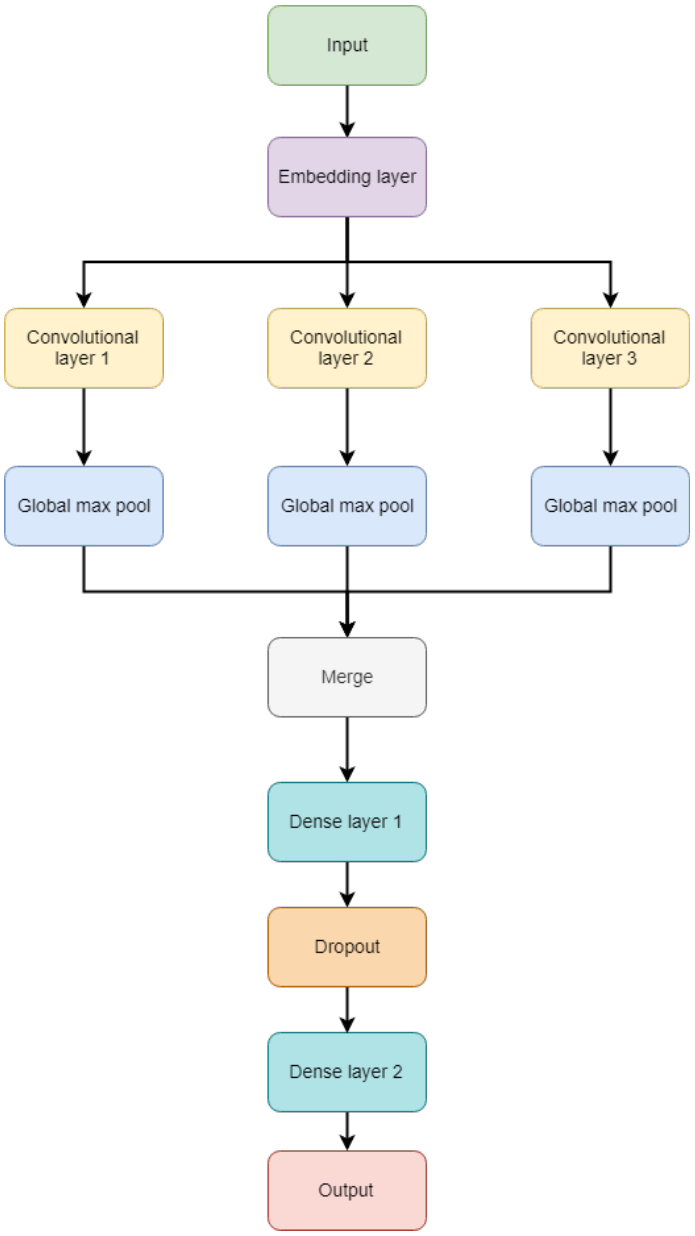


Fig. 21. Suggested Convolutional Neural Network model with 3 convolutional layers

The parameter values mentioned below in each component of the model were for the initial model. The final parameter tuning had been evaluated based on the different experiments in this study.

A. Input

The processed US Census Name Gender dataset [13] had been used with preprocesing as explained in the Section IV.

B. Embedding Layer

An embedding was a mapping of a discrete categorical variable to a vector of continuous numbers. In the context of Neural Networks, embeddings were low-dimensional and learned continuous vector representations of discrete variables. Neural Network embeddings were useful because they could reduce the dimensionality of categorical variables and meaningfully represented categories in the transformed space. In conclusion, the embedding layer turned positive integers (indexes) into dense vectors of fixed size.

Embedded layer parameters were as follows.

- Input Length (the length of input sequences)- equal to size of dense vector in input.
- Input Dimensions (the size of the vocabulary in the text data) - 27 (1 to 26 for alphabet index and 0 padding index)
- Output Dimensions (the size of the vector space in which words will be embedded) - 32 (embedded layer size)
- Weight Initialization Function - uniform

C. Convolutional Layers

3 convolutional layers were defined with the following parameters.

- Filter sizes - 1,2,3 initially
- Activation Function - Tanh
- Number of Filters - 32 (equal to the embedded layer output dimension)
- Sub Sample Length - 1
- Weight Initialization Function - Glorot uniform
- Border Mode - valid
- Bias - true

D. Max Pooling Layers

Global max pooling layer was applied to each convolutional layer.

E. Merge

Merge function was applied to get the sum of the 3 global max pooling layers.

F. Dense Layer (Fully Connected Layer)

Fully connected layer 1 was configured with output dimensions which equal to the output dimension of the embedded layer and activation function Sigmoid.

Fully connected layer 2 was configured with output dimension 1 and activation function Sigmoid to generate the final output.

G. Dropout Layer

Dropout layer with dropout rate 0.2 was applied.

H. Output

Output 1 (Male) or 0 (Female).

VI. Methodology

In this thesis, different experiments were carried out to find the optimum model. The objectives of each experiment and the conclusion can be found under the each experiment.

A. Different input lengths

1) Objectives: To check the performance of the initial model compared to the length of the input name vector and to find the minimum length of the input which has the minimum training time without losing the performance of the rest of the experiments in order to reduce the training time of the experiments. And to check whether the model performance depends on the input name length.

TABLE III
Number of records and the percentage of the number of records compared to the total records, for the given maximum length of names in US Census dataset [13]

Maximum length of names	**Total records**	**Percentage of total records (%)**
15	95654	100
14	95620	100
13	95556	100
12	95420	100
11	95196	100
10	94583	99
9	92842	97
8	87511	91
7	73685	77
6	48744	51
5	22118	23
4	6137	6

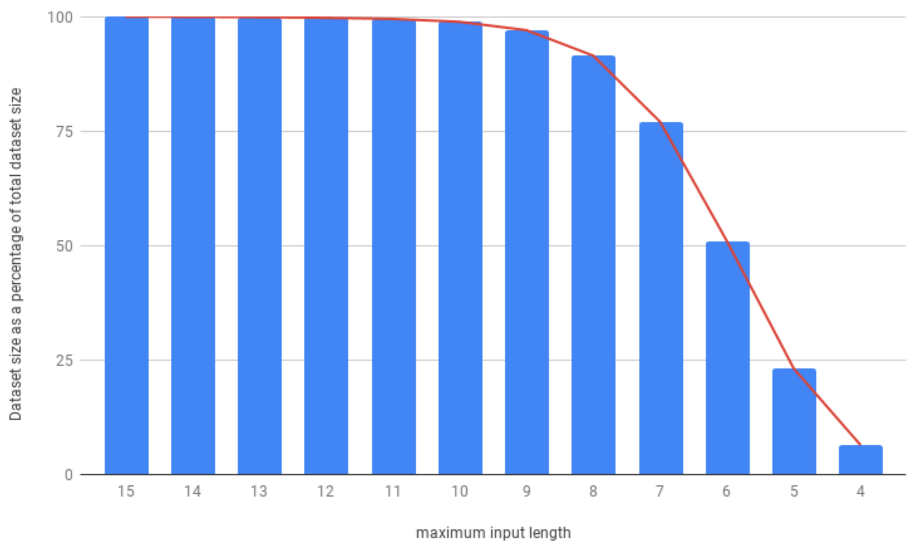


Fig. 22. Percentage of the dataset size of the selected maximum length of the name compared to the total dataset size in US Census dataset [13]

The length of the maximum name in the US Census dataset was 15. The model was trained for different input lengths and accuracy, loss, training time were recorded in each training. The distribution of the records was compared to the length of the name illustrated in Table III and Figure 22. Output size of the embedded layer was set to 32 to reduce the training time and the number of the epoch was set to 15.

According to the results (Figure 23), performance started decreasing after the input length was 9. According to table III, dataset size started decreasing gradually after input length was 9. Therefore, the performance did not depend on the input length but it depended on the dataset size.

TABLE IV
TRAINING TIME, VALIDATION ACCURACY AND LOSS FOR DIFFERENT INPUT LENGTHS OF THE MODEL IN FIGURE 21

Input length	Training time (Hours:Minutes:Seconds)	Loss	Accuracy
15	0:11:38	0.3130	0.8750
14	0:11:00	0.2767	0.8850
13	0:11:12	0.2790	0.8850
12	0:10:20	0.2819	0.8830
11	0:10:15	0.2790	0.8847
10	0:10:00	0.2835	0.8825
9	0:09:58	0.2820	0.8827
8	0:09:56	0.2843	0.8814
7	0:07:46	0.3048	0.8688
6	0:05:19	0.3423	0.8528
5	0:02:27	0.3952	0.8284
4	0:00:44	0.4441	0.8101

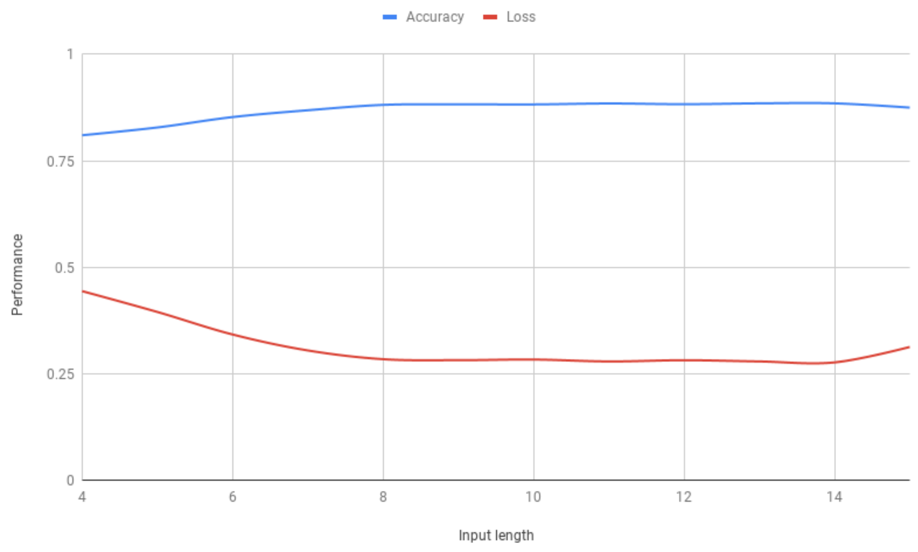


Fig. 23. Accuracy and loss results for different input lengths of the model in Figure 21

As of the Table IV and Figure 24, the training time difference between input length 15 and 9 was around 2 minutes. Therefore, input length 9 was selected for the rest of the experiments, since there was no much performance difference between input length 9 and 15, but it saved training time.

2) Conclusion: The performance of the model does not depend on the input name length. But the dataset size affects the performance. The large dataset provides the maximum performance gain. The training time varies based on the input name vector length. Input name vector length 9 provides good performance and less training time when compared to the higher input name vector lengths.

B. Different filter sizes

1) Objective: To find the filter size or sizes which provide the maximum performance.

The model ran for different filter sizes with the following hyper parameters.

- Epoch: 15
- Batch Size: 100
- Loss Function: Binary Cross Entropy
- Optimizer name: Adam
- Data set split to 80%, 10%, 10% as Training, Validation and Testing respectively

According to the table V and figures 25 , 26, for the single filters, filter size 7 gave the maximum training accuracy (0.8821) and validation accuracy (0.8728) with minimum training loss (0.2831) and validation loss (0.3059). When considering dual

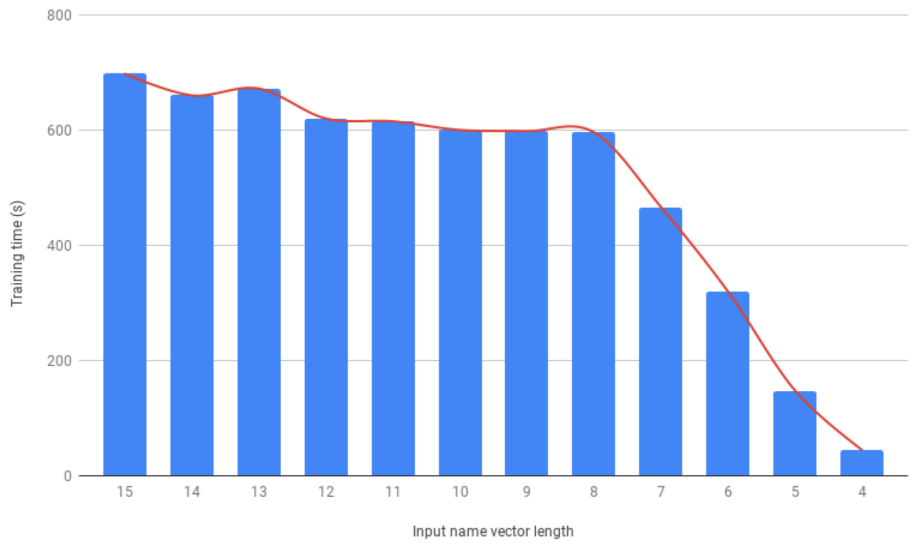


Fig. 24. Training time compared to the input name vector length of the model in Figure 21

TABLE V
TRAINING LOSS AND ACCURACY, VALIDATION LOSS AND ACCURACY WITH LOSS DIFFERENCE AND ACCURACY DIFFERENCE FOR DIFFERENT FILTER SIZES IN CONVOLUTIONAL LAYERS OF THE MODEL IN FIGURE 21

Filter Sizes	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Loss Difference (validation-training)	Accuracy Difference (training-validation)
1	0.6100	0.6732	0.6000	0.6906	-0.0100	-0.0174
2	0.3550	0.8443	0.3563	0.8440	0.0013	0.0003
3	0.3159	0.8661	0.3137	0.8649	-0.0022	0.0012
4	0.2514	0.8746	0.3143	0.8676	0.0183	0.0070
5	0.2923	0.8772	0.3067	0.8718	0.0144	0.0144
6	0.2895	0.8786	0.3121	0.8684	0.0226	0.0226
7	**0.2831**	**0.8821**	**0.3059**	**0.8728**	0.0228	0.0228
1,2	0.3423	0.8520	0.3385	0.8487	-0.0038	-0.0033
2,3	0.3032	0.8723	0.3130	0.8663	0.0098	0.0060
3,4	0.2683	0.8884	**0.2779**	0.8835	0.0096	0.0049
4,5	0.2535	0.8958	0.2850	**0.8835**	0.0315	0.0123
5,6	0.2515	0.8959	0.2785	0.8816	0.0270	0.0143
6,7	**0.2428**	**0.8989**	0.2976	0.8760	0.0548	0.0229
1,2,3	0.2817	0.8826	0.3026	0.8677	0.0209	0.0149
2,3,4	0.2348	0.9024	0.2843	0.8811	0.0495	0.0213
3,4,5	0.2178	0.9110	**0.2638**	**0.8971**	0.0460	0.0139
4,5,6	**0.2042**	**0.9163**	0.2766	0.8911	0.0724	0.0252
1,2,3,4	0.2605	0.8923	0.2880	0.8803	0.0275	0.0120
2,3,4,5	0.2361	0.9024	0.2695	**0.8899**	0.0334	0.0125
3,4,5,6	**0.2219**	**0.9083**	**0.2686**	0.8898	0.0467	0.0185

filters, filter size 6,7 outperformed others by both training accuracy (0.8989) and training loss (0.2428). But filter size 3,4 and 4,5 gave the maximum validation accuracy (0.8835) and filter size 3,4 gave the minimum loss (0.2779). From tri filters, even though filter size 4,5,6 gave the maximum training accuracy (0.9163) and minimum training loss (0.2042) when considering both training and validation accuracy, loss, filter size 3,4,5 performed well with less over-fitted results (accuracy: 0.8971, loss: 0.2638). Among the four filters, filter size 3,4,5,6 performed well.

Selected filter sizes based on above results were listed in table VI. According to the selected filters table VI, filter size 3,4,5 outperformed others in both validation accuracy, loss and difference between the training and validation accuracy and loss. Filter size 3,4,5 less over-fitted compared to the other filter sizes. Therefore, filter size 3,4,5 was selected as the filter size for the rest of the experiments.

2) Conclusion: Filter size 3,4,5 provides less over-fitted maximum performance compared to other filters. Increasing the number of filter sizes does not increase the performance as expected.

TABLE VI
SELECTED RESULTS FROM THE TABLE V FOR DIFFERENT FILTER SIZES

Filter Sizes	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Accuracy Difference (training-validation)	Loss Difference (validation-training)
3,4	0.2683	0.8884	0.2779	0.8835	0.0096	0.0049
4,5	0.2535	0.8958	0.2850	0.8835	0.0315	0.0123
3,4,5	0.2178	0.9110	**0.2638**	**0.8971**	**0.0460**	**0.0139**
4,5,6	**0.2042**	**0.9163**	0.2766	0.8911	0.0724	0.0252
3,4,5,6	0.2219	0.9083	0.2686	0.8898	0.0467	0.0185

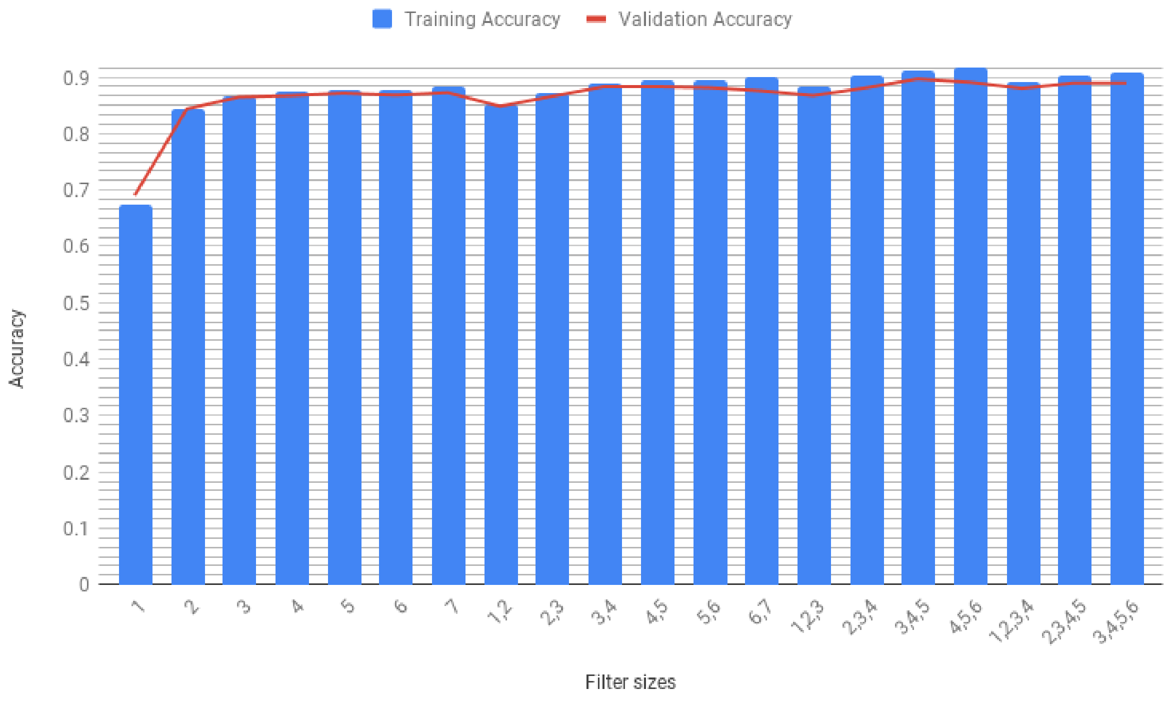


Fig. 25. Accuracy difference between different filter sizes in convolutional layer of the model in Figure 21

C. *Different embedded layer sizes*

1) Objective: To check the effect of the embedded layer size for the model performance.

TABLE VII
TRAINING AND VALIDATION PERFORMANCE ALONG WITH THE TRAINING TIME BETWEEN DIFFERENT EMBEDDED LAYER SIZES OF THE MODEL IN FIGURE 21

Embedded Layer Size	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Training Time (seconds)
24	0.9000	0.2402	0.8862	0.2776	439
32	0.9110	0.2178	0.8971	**0.2638**	410
64	0.9306	0.1661	0.8915	0.2868	446
128	0.9505	0.1196	0.8943	0.2863	556
256	**0.9600**	**0.0967**	0.8996	0.2917	1001
512	0.9560	0.1080	**0.9000**	0.2937	2474

According to the table VII, embedded size 256, gave the maximum training accuracy and minimum loss of 0.9600 and 0.0967 respectively which was quite good performance compared to the other results. However, embedded size 512 gave the maximum validation accuracy of 0.9000 and embedded size 32 gave the minimum validation loss of 0.2638.

According to the figures 27, 28, embedded size 32 gave the less over-fitted results over others. Also, when considering the training time (Fig. 29), embedded size 32 showed less training time. If embedded size increases the training time also increases exponentially. Therefore, even though a large embedded size increased the performance by a small amount, compared to the training time it was not much effective. As a conclusion, embedded size 32 was better than others.

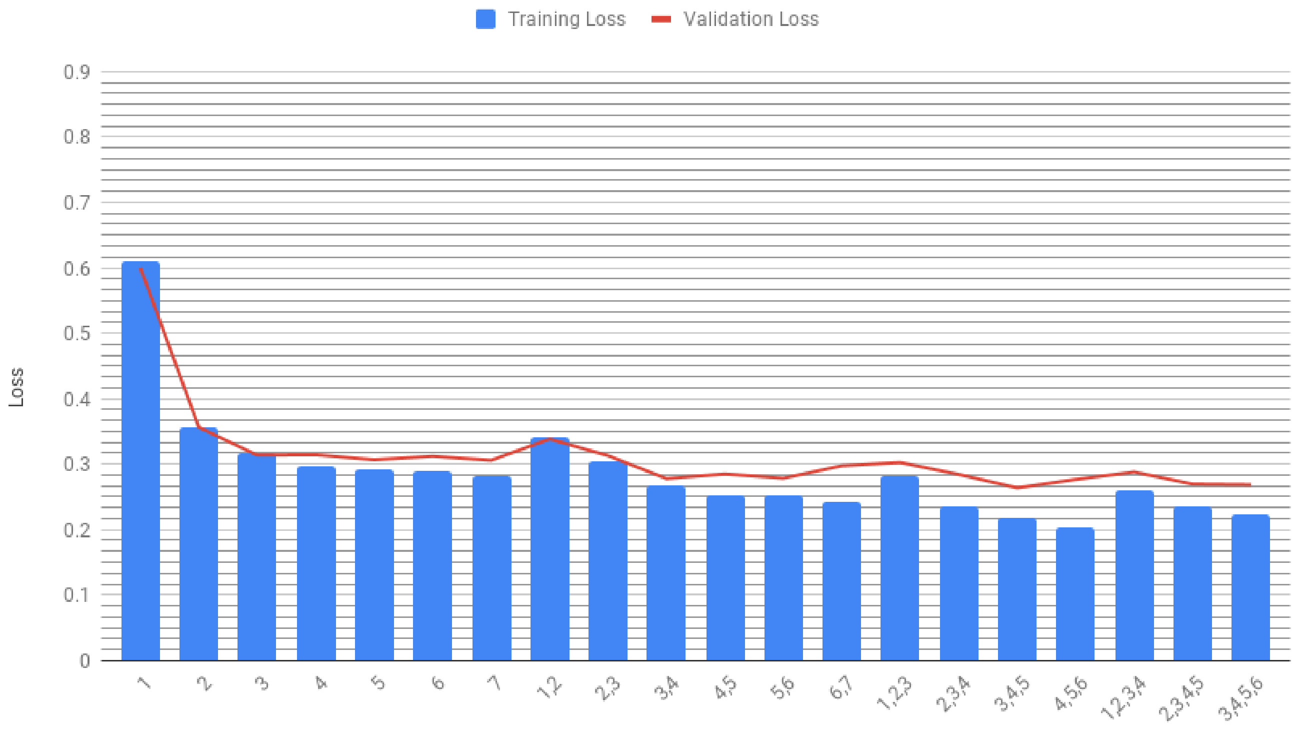


Fig. 26. Loss difference between different filter sizes in convolutional layer of the model in Figure 21

2) Conclusion: Embedded size 32 shows the maximum performance with less over-fitted results and comparably low training time for the model.

D. Different activation functions in different layers

1) Objective: To find the activation function setup which gives the maximum results.

Results of different activation functions in different layers with using 2 Dense (Fully Connected) layers shown in the table VIII and with 3 Dense layers shown in table IX.

TABLE VIII
TRAINING AND VALIDATION PERFORMANCE BETWEEN DIFFERENT ALGORITHMS IN DIFFERENT LAYERS WITH 2 DENSE LAYER MODEL IN FIGURE 21

Convolutional Layers	**Dense1**	**Dense2**	**Training Accuracy**	**Training Loss**	**Validation Accuracy**	**Validation Loss**
Relu	Sigmoid	Sigmoid	0.9078	0.2256	0.8878	0.2747
Softmax	Sigmoid	Sigmoid	0.9016	0.2390	0.8916	0.2684
Tanh	Sigmoid	Sigmoid	0.9110	0.2178	0.8971	0.2638
Tanh	Softmax	Sigmoid	0.9064	0.2374	0.8875	0.2803
Tanh	Tanh	Sigmoid	0.9156	0.2105	0.8876	0.2756
Tanh	Relu	Sigmoid	0.9130	0.2101	0.8888	0.2704
Tanh	Sigmoid	Linear	0.8332	0.4058	0.8598	0.3500
Tanh	Sigmoid	Relu	0.8463	0.3851	0.8599	0.3613
Tanh	Sigmoid	Softmax	0.5000	7.9712	0.3634	10.1489
Tanh	Relu	Linear	0.7725	0.4711	0.8171	0.4310
Tanh	Relu	Softmax	0.3634	10.1481	0.3710	10.0273
Tanh	Relu	Relu	0.7984	0.4775	0.8345	0.4111

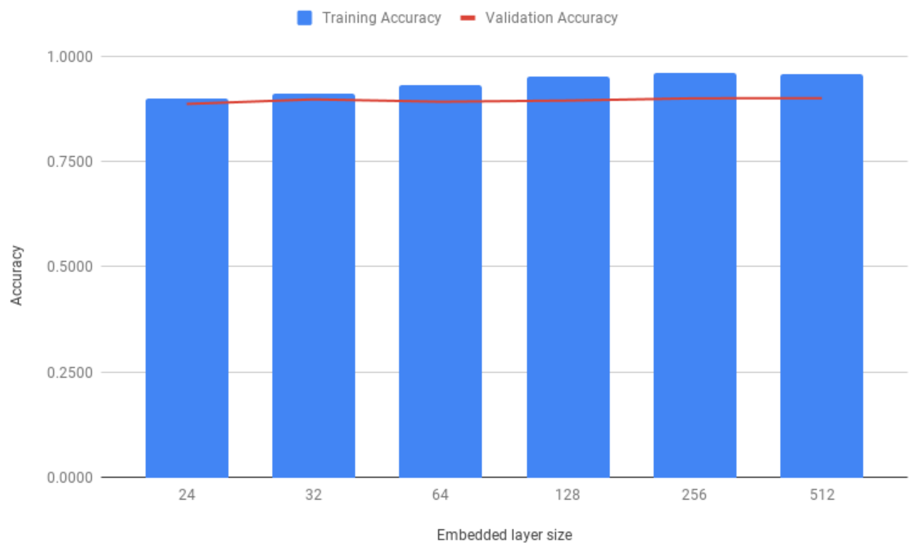


Fig. 27. Accuracy difference between different embedded layer sizes of the model in Figure 21

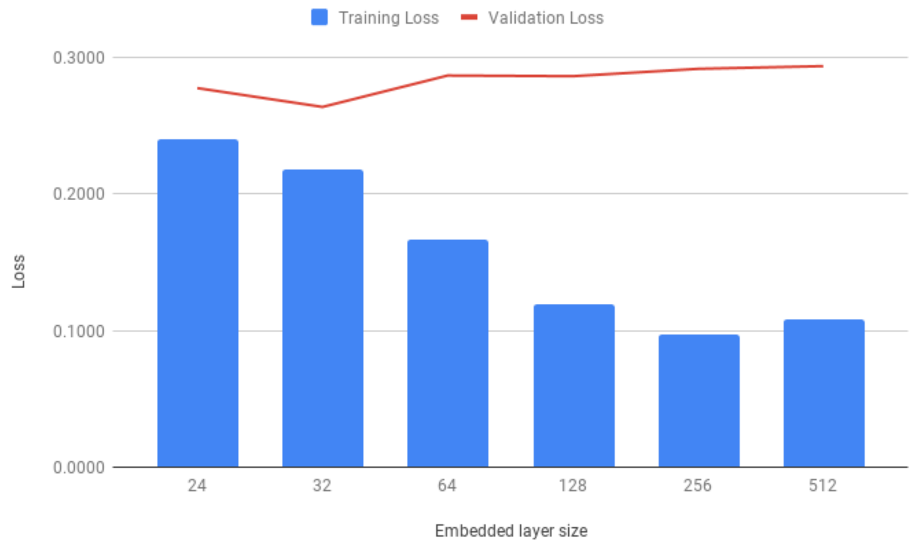


Fig. 28. Loss difference between different embedded layer sizes of the model in Figure 21

2) Activation function Relu in Convolutional layers: The activation function was changed as Relu in each Convolutional layer and the accuracy and loss were recorded.

3) Activation function Softmax in Convolutional layers: The activation function was changed as Softmax in each Convolutional layer and the accuracy and loss were recorded.

4) Activation function Tanh in Convolutional layers: The Tanh activation function was used for the models in previous experiments.

When comparing 3 different activation functions (Tanh, Relu, Softmax) for Convolutional layers, the Tanh activation function outperformed others by performance (Figure 30). Therefore, the activation function Tanh was selected as the Convolutional layer activation function for the rest of the experiments.

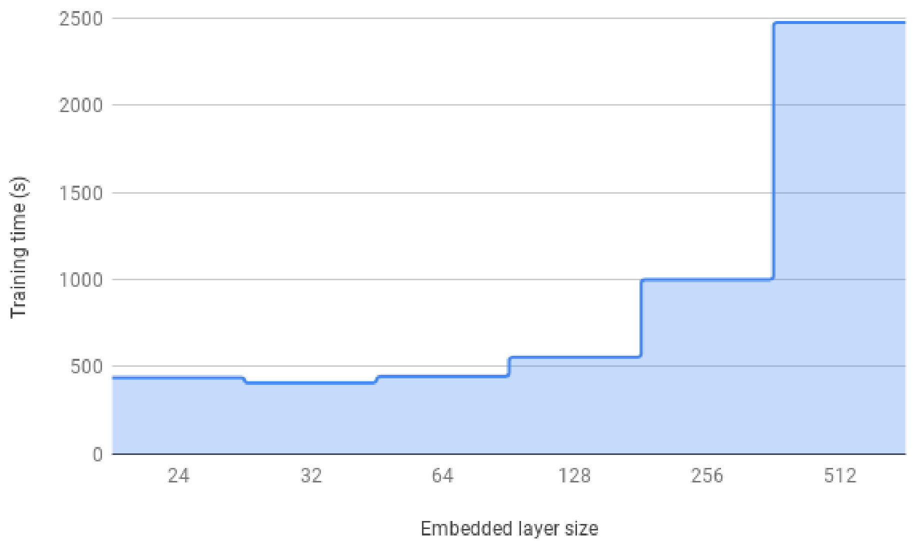


Fig. 29. Training time difference between different embedded layer sizes of the model in Figure 21

TABLE IX
TRAINING AND VALIDATION PERFORMANCE BETWEEN DIFFERENT ALGORITHMS IN DIFFERENT LAYERS WITH 3 DENSE LAYER MODEL IN FIGURE 33

Convolutional Layers	Dense1	Dense2	Dense3	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Tanh	Relu	Relu	Sigmoid	0.9118	0.2150	0.8905	0.2686
Tanh	Relu	Relu	Relu	0.8510	0.3779	0.8505	0.3512
Tanh	Sigmoid	Sigmoid	Sigmoid	0.9114	0.2185	0.8841	0.2844

5) Activation function Tanh in Convolutional layers, Dense 1 Softmax and Dense 2 Sigmoid: Dense 1 activation function was changed to Softmax with Tanh in Convolutional layers and Sigmoid in Dense2 as activation functions.

6) Activation function Tanh in Convolutional layers, Dense 1 Tanh and Dense 2 Sigmoid: Dense 1 activation function was changed to Tanh with Tanh in Convolutional layers and Sigmoid in Dense2 as activation functions.

7) Activation function Tanh in Convolutional layers, Dense 1 Relu and Dense 2 Sigmoid: Dense 1 activation function was changed to Relu with Tanh in Convolutional layers and Sigmoid in Dense2 as activation functions.

8) Activation function Tanh in Convolutional layers, Dense 1 Sigmoid and Dense 2 Linear: Ususally, the Linear activation function is used in the final fully connected layer. Therefore, Dense2 activation function was set as Linear where Tanh was set as Convolutional layer activation function and Sigmoid in Dense 1.

The recorded training accuracy was around 0.83 and validation accuracy was around 0.85 which was less than the model with activation function Sigmoid in Dense2.

9) Activation function Tanh in Convolutional layers, Dense 1 Sigmoid and Dense 2 Relu: Dense 2 activation function was changed to Relu with Tanh in Convolutional layers and Sigmoid in Dense 1 as activation functions.

10) Activation function Tanh in Convolutional layers, Dense 1 Sigmoid and Dense 2 Softmax: Dense 2 activation function was changed to Softmax with Tanh in Convolutional layers and Sigmoid in Dense 1 as activation functions.

11) Activation function Tanh in Convolutional layers, Dense 1 Relu and Dense 2 Linear: Dense 2 activation function was changed to Linear with Tanh in Convolutional layers and Relu in Dense 1 as activation functions. Training and validation accuracy were around 0.81 and training was not smooth as in Figure 31.

12) Activation function Tanh in Convolutional layers, Dense 1 Relu and Dense 2 Softmax: Dense 2 activation function was changed to Softmax with Tanh in Convolutional layers and Relu in Dense 1 as activation functions. Training accuracy did not increase beyond 0.4 and training loss was around 10 (Figure 32).

When comparing different algorithms in Dense layers (Table VIII), ”Activation function Tanh in Convolutional layers, Dense 1 Relu and Dense 2 Sigmoid” model gave the maximum training accuracy and minimum training loss. However, when considering both training and validation performance, the model ”Activation function Tanh in Convolutional layers, Dense 1 Sigmoid and Dense 2 Sigmoid” showed the best results which were less over-fitted when compared to the model ”Activation function Tanh in Convolutional layers, Dense 1 Relu and Dense 2 Sigmoid” .

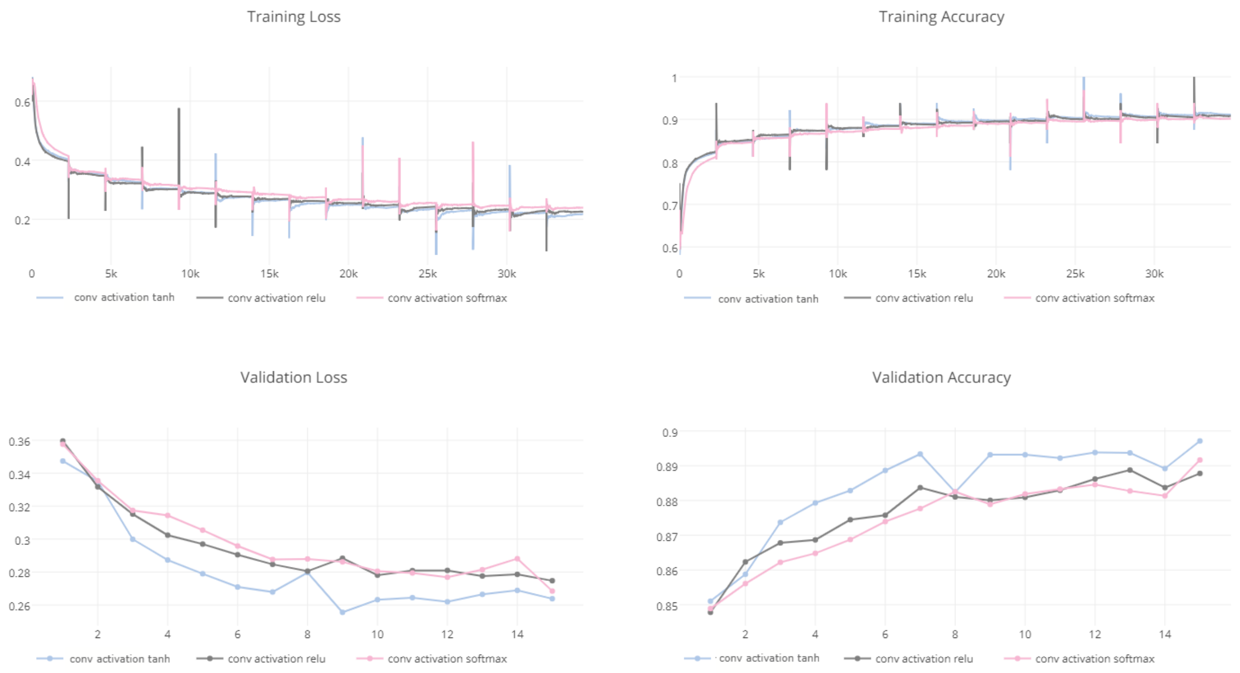


Fig. 30. Comparison of activation functions Tanh, Relu, Softmax for Convolutional layers of the model in Figure 21

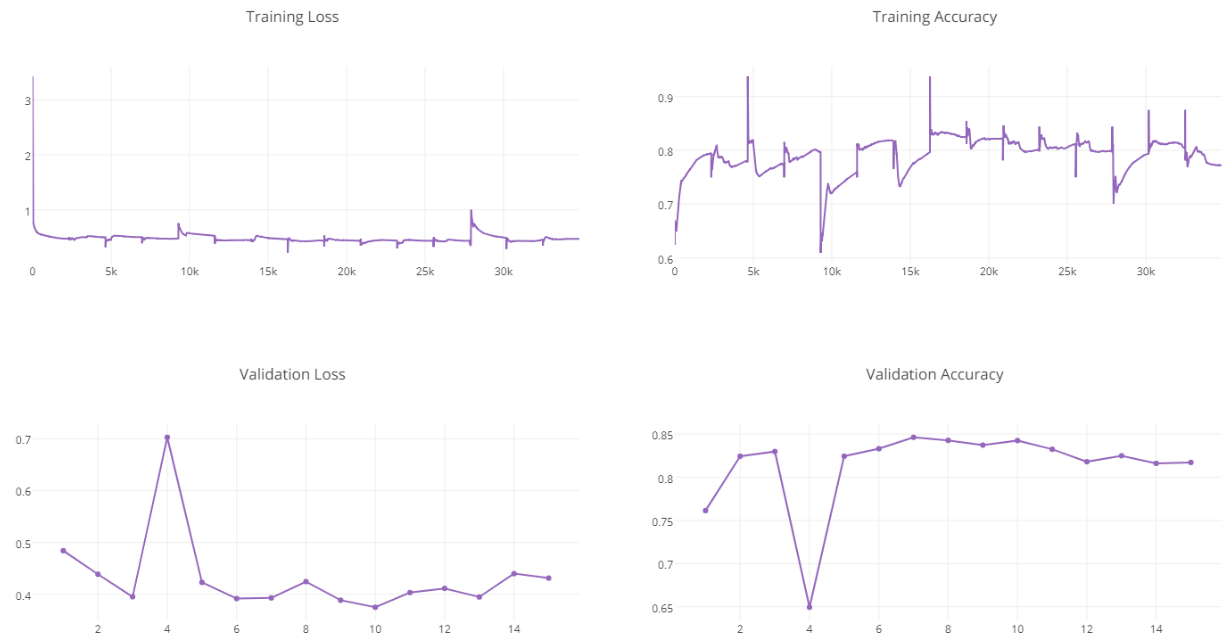


Fig. 31. Performance of activation function Tanh in Convolutional layers, Dense 1 Relu and Dense 2 Linear of the model in Figure 21

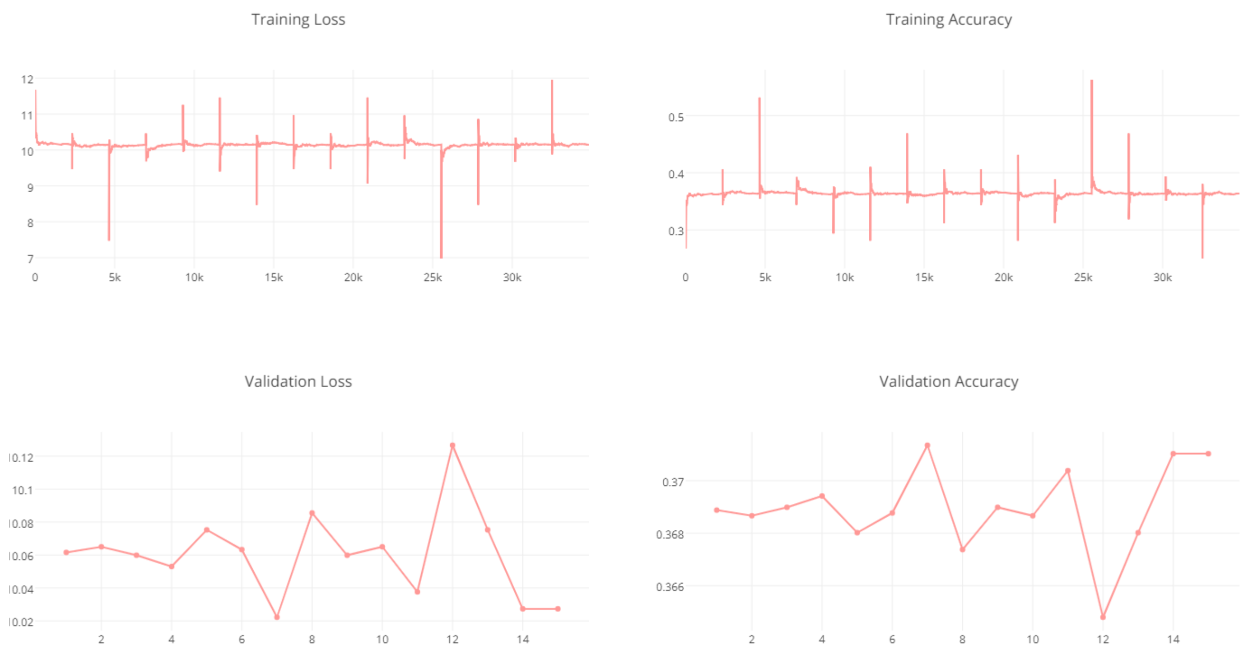


Fig. 32. Performance of activation function Tanh in Convolutional layers, Dense 1 Relu and Dense 2 Softmax of the model in Figure 21

A model with 3 Dense layers (Figure 33) was used in the next few experiments in order to compare the performance with 2 Dense layer model. The Dense layer output size is shown as below.

- Dense 1 : 32
- Dense 2 : 16
- Dense 3 : 1

13) Activation function Tanh in Convolutional layers, Dense 1 Relu, Dense 2 Relu, Dense 3 Sigmoid: Dense 3 activation function was set to Sigmoid with Tanh in Convolutional layers, Relu in Dense 1 and Dense 2 as activation functions.

14) Activation function Tanh in Convolutional layers, Dense 1 Sigmoid, Dense 2 Sigmoid, Dense 3 Sigmoid: Activation function was set to Sigmoid in all Dense layers with Tanh in Convolutional layers. This setup performed well but results were bit lower compared to the setup ”Activation function Tanh in Convolutional layers, Dense 1 Relu, Dense 2 Relu, Dense 3 Sigmoid”.

15) Activation function Tanh in Convolutional layers, Dense 1 Relu, Dense 2 Relu, Dense 3 Relu: Activation function was set to Relu in all Dense layers with Tanh in Convolutional layers. Training was not smooth as shown in figure 34.
Adding more dense layers did not increase the performance but it increased the complexity of the model according to the results in the table IX.

The model with 2 dense layer and activation function as Sigmoid outperformed the other models.

16) conclusion: In this experiment, 2 models were considered with 2 Dense layers (initial model) and 3 Dense layers. According to the results, adding more Dense layers does not increase the model performance, but it adds additional complexity to the model. The best performing model contains 2 Dense layers with Tanh as activation function in Convolutional layers and Sigmoid as activation function in Dense layers.

E. First n characters and last n characters

1) Objective: To check whether it is possible to reduce the input size without degrading the performance.

As of the table X, the model gave maximum performance (accuracy around 0.85) for first 2 and last 2 character input. But it showed less performance than using the whole name as an input to the model.

2) Conclusion: Reducing the input size of the input by using first n, last n characters does not increase the performance.

F. 3 Fold cross validation

1) Objective: To cross validate the results.

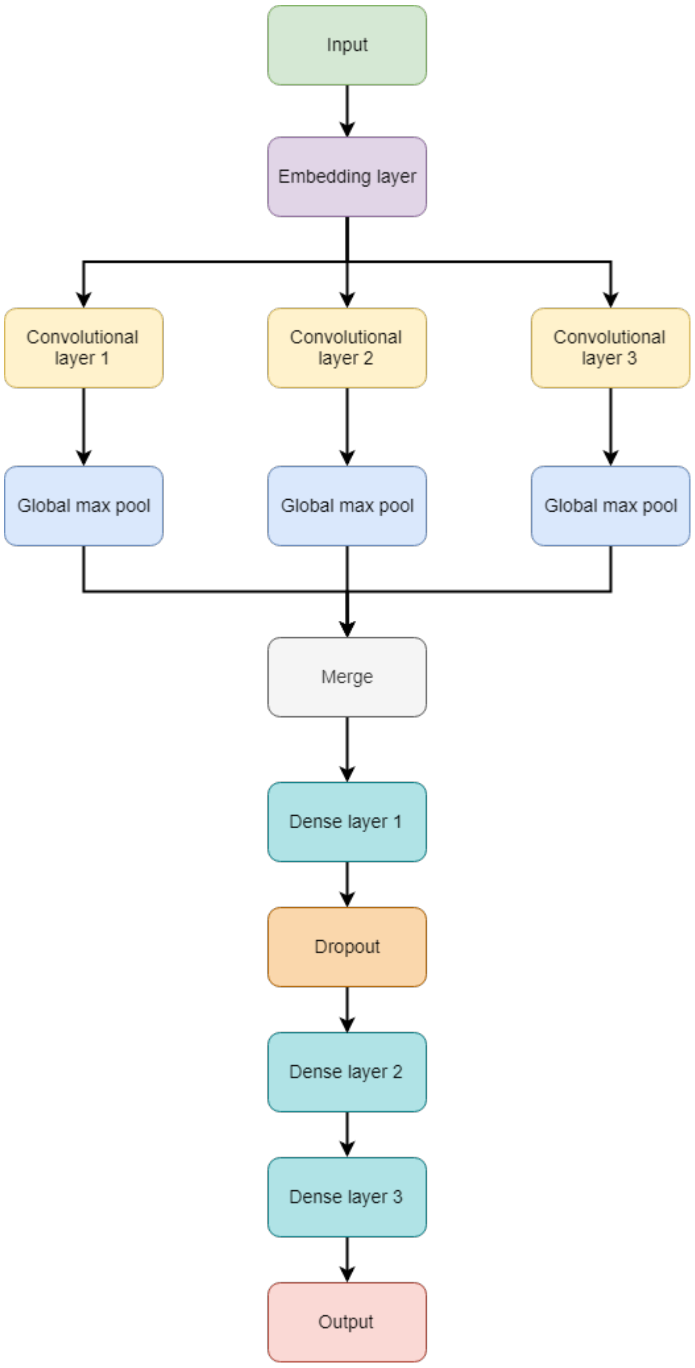


Fig. 33. CNN model with 3 Dense layers

The final model (Figure 21) with the following parameters tuning was checked with the 3 Fold cross-validation.

Activation Functions

- Convolutional layers : Tanh
- Dense layers : Sigmoid

Other Parameters

- Convolutional layer filter sizes : 3,4,5
- Embedded layer size : 32
- Dense 1 output size : 32

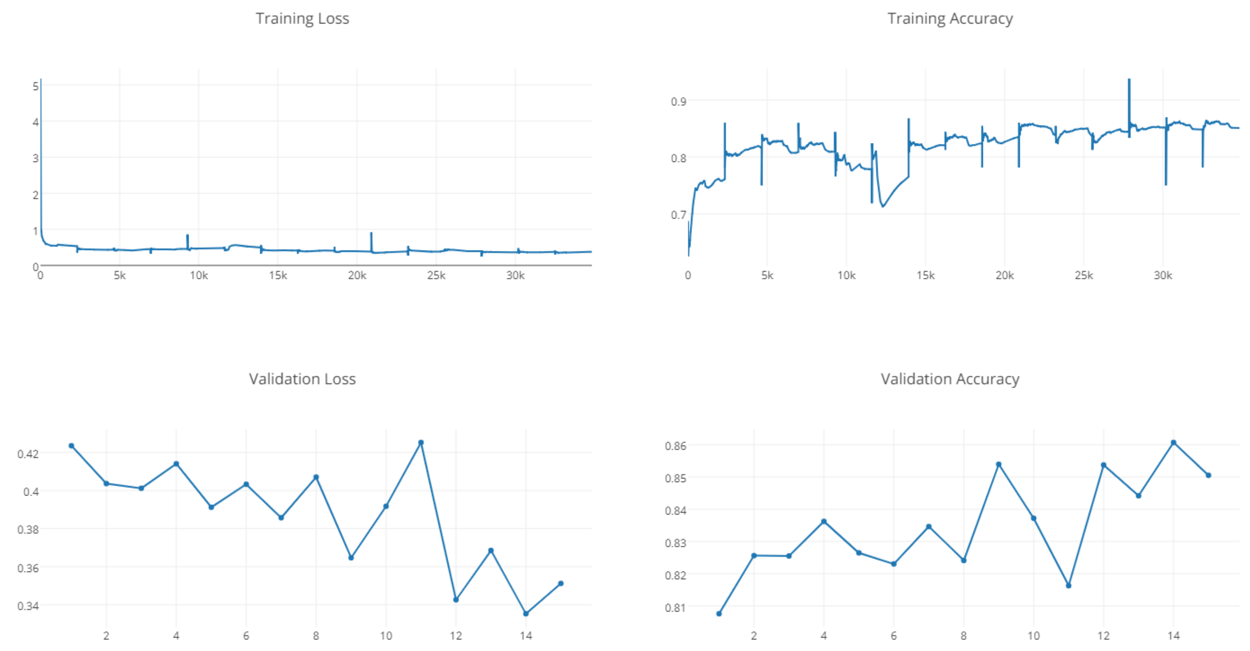


Fig. 34. Performance of activation function Tanh in Convolutional layers, Dense 1 Relu, Dense 2 Relu and Dense 3 Relu of the model in Figure 33

TABLE X
VALIDATION ACCURACY AND LOSS PERFORMANCE FOR FIRST N CHARACTERS AND LAST N CHARACTERS BASED FEATURES OF THE MODEL IN FIGURE 21

First n characters (n=)	Last n characters (n=)	Accuracy	Loss
1	2	0.83	0.37
2	2	**0.85**	**0.36**
2	1	0.81	0.42

- Dense 2 output size : 1
- Dropout : 0.2
- Number of epoch : 15
- Batch size : 32
- Loss function : Binary cross entropy
- Optimizer : Adam

TABLE XI
TRAINING AND VALIDATION PERFORMANCE WITH 3 FOLD CROSS VALIDATION OF THE FINAL MODEL IN FIGURE 21

	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Iteration 1	0.9087	0.2220	0.8902	0.2755
Iteration 2	0.9069	0.2260	0.8867	0.2805
Iteration 3	0.9097	0.2179	0.8814	0.2858
Average	0.9084	0.2223	0.8861	0.2806

Figure 35 showed the results of the 3 Fold cross-validation. Training data size was 61894 and validation data size was 30947 in each iteration. Final results were almost similar in all the iterations. The average training accuracy, training loss, validation accuracy, validation loss were 0.9084, 0.2223, 0.8861, 0.2806 respectively (Table XI).

2) Conclusion: Average results are similar in all the iterations.

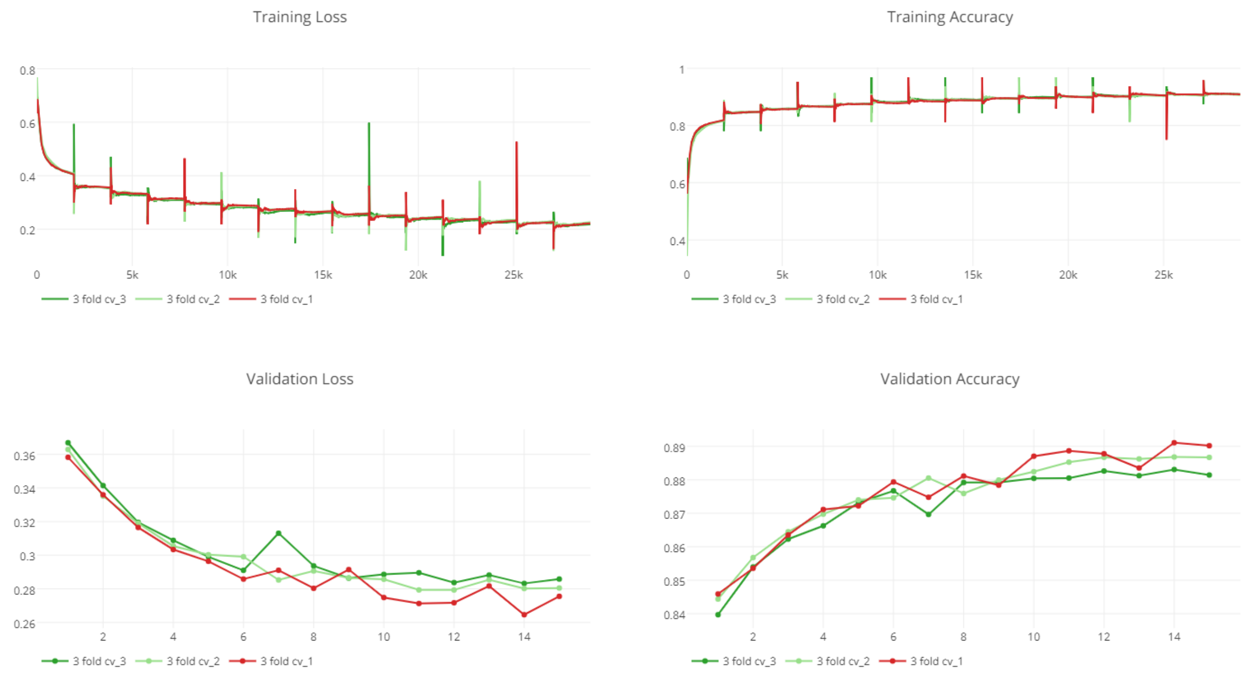


Fig. 35. Performance of 3 Fold cross validation of the model in Figure 21

VII. Conclusions

A. Summary

This thesis defines a gender classification of people, based on the first name, with character level 1D parallel Convolutional Neural Network (CNN). The final model contains 3 parallel Convolutional layers, 2 Fully Connected layers and 1 Dropout layer (Figure 21). The model tests different scenarios like changing the input length, changing the model, changing activation functions and changing the input. According to the experiments, Tanh as activation function for Convolutional layers and Sigmoid activation function for Fully Connected layers outperform the other models. The model does not depend on the input length, but it depends on the dataset size. Usually, CNN models need a large dataset. The final model gives an accuracy of around 0.89 which outperform the traditional methods. Following are the parameters tuning and activation functions used in the final model.

Activation Functions

- Convolutional layers : Tanh
- Dense layers : Sigmoid

Other Parameters

- Convolutional layer filter sizes : 3,4,5
- Embedded layer size : 32
- Dense 1 output size : 32
- Dense 2 output size : 1
- Dropout : 0.2
- Number of epoch : 15
- Batch size : 32
- Loss function : Binary cross entropy
- Optimizer : Adam

B. Future Work

In this research, only the US census name gender dataset ([13]) was used for training and validation. The model should test other datasets.

On the other hand, it is needed to check the model performance by providing the heuristic knowledge along with the name as an input.

VIII. Appendices

A. *Yaml configuration file for load in to Deep Leraning Studio*

```
data:
  dataset: {name: uscensus-9, samples: 92842, type: private}
  datasetLoadOption: batch
  kfold: 1
  mapping:
    gender:
      options: {Normalization: false, Scaling: 1}
      port: OutputPort0
      shape: ''
      type: Numeric
    name:
      options: {Normalization: false, Scaling: 1}
      port: InputPort0
      shape: ''
      type: Array
  numPorts: 1
  samples: {split: 4, test: 9284, training: 74273, validation: 9284}
  shuffle: true
model:
  connections:
  - {source: GlobalMaxPooling1D_4, target: merge_2}
  - {source: GlobalMaxPooling1D_1, target: merge_2}
  - {source: GlobalMaxPooling1D_9, target: merge_2}
  - {source: Embedding_1, target: Convolution1D_1}
  - {source: Embedding_1, target: Convolution1D_6}
  - {source: Embedding_1, target: Convolution1D_11}
  - {source: Convolution1D_6, target: GlobalMaxPooling1D_4}
  - {source: Convolution1D_11, target: GlobalMaxPooling1D_9}
  - {source: Convolution1D_1, target: GlobalMaxPooling1D_1}
  - {source: Dropout_3, target: Dense_15}
  - {source: Input_1, target: Embedding_1}
  - {source: merge_2, target: Dense_13}
  - {source: Dense_15, target: Output_1}
  - {source: Dense_13, target: Dropout_3}
  layers:
  - args: {}
    class: Input
    name: Input_1
    x: 322
    y: 20
  - args: {}
    class: Output
    name: Output_1
    x: 242
    y: 964
  - args: {input_dim: '28', input_length: '9', output_dim: '32'}
    class: Embedding
    name: Embedding_1
    x: 322
    y: 130
  - args: {activation: tanh, filter_length: '3', nb_filter: '32'}
    class: Convolution1D
    name: Convolution1D_1
    x: 30
    y: 279
  - args: {}
    class: GlobalMaxPooling1D
    name: GlobalMaxPooling1D_1
    x: 29
    y: 411
  - args: {activation: sigmoid, output_dim: '32'}
    class: Dense
    name: Dense_13
    x: 241
    y: 646
  - args: {p: '0.2'}
    class: Dropout
    name: Dropout_3
    x: 240
    y: 747
  - args: {activation: sigmoid, output_dim: '1'}
    class: Dense
    name: Dense_15
    x: 241
    y: 857
  - args: {activation: tanh, filter_length: '5', nb_filter: '32'}
    class: Convolution1D
    name: Convolution1D_6
    x: 327
    y: 280
  - args: {}
    class: GlobalMaxPooling1D
    name: GlobalMaxPooling1D_4
    x: 329
    y: 412
  - args: {}
    class: merge
    name: merge_2
    x: 238
    y: 546
  - args: {activation: tanh, filter_length: '4', nb_filter: '32'}
    class: Convolution1D
    name: Convolution1D_11
    x: 607
    y: 279
  - args: {}
    class: GlobalMaxPooling1D
    name: GlobalMaxPooling1D_9
    x: 608
    y: 411
params:
  advance_params: true
  batch_size: 32
  is_custom_loss: false
  loss_func: binary_crossentropy
  num_epoch: 15
  optimizer: {name: Adam}
project: CNN Gender Classification
```

Hosted version of the model script can be found in the following url.
https://github.com/maduraPradeep/uscensus-name-gender-for-cnn/blob/master/model/CNNGenderClassification.yaml

B. Data preprocess source code - Github

Data preprocess source code is hosted on Github which can be found in the following url.
https://github.com/maduraPradeep/uscensus-name-gender-for-cnn
In order to build the source, it is required to install nodejs in the computer. Then steps below should be followed.

1) Clone the repository
2) Run npm install
3) Run node uscensusDataProcessor.js to generate the train.csv file

C. Preprocessed dataset for training - Github

The preprocessed dataset which used for the training is hosted on Github which can found in the following url.
https://github.com/maduraPradeep/uscensus-name-gender-for-cnn/blob/master/train.zip

REFERENCES

[1] F. Vázquez. Deep learning made easy with deep cognition. [Online]. Available: https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351

[2] Wikipedia website. [Online]. Available: https://www.wikipedia.org

[3] Researchgate website. [Online]. Available: https://www.researchgate.net

[4] Slideshare website. [Online]. Available: https://www.slideshare.net

[5] Y. Chen, "Convolutional neural network for sentence classification."

[6] W. Liu and D. Ruths, "What's in a name? using first names as features for gender inference in twitter."

[7] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network."

[8] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "Learning semantic representations using convolutional neural networks for web search."

[9] X. H. Wen tau Yih and C. Meek, "Semantic parsing for singlerelation question answering."

[10] Y. Kim, "Convolutional neural networks for sentence classification."

[11] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences."

[12] E. G. Nal Kalchbrenner and P. Blunsom, "A convolutional neural network for modelling sentences."

[13] The us census website. [Online]. Available: https://www.census.gov

[14] S. Modak and A. C. Mondal, "A comparative study of classifiers' performance for gender classification."

[15] A. A. Septiandri, "Predicting the gender of indonesian names."

[16] Deep cognition website. [Online]. Available: https://deepcognition.ai

[17] U. M. Dan Ciresan and J. Schmidhuber, "Multi-column deep neural networks for image classification. in computer vision and pattern recognition (cvpr)," p. 3642–3649.

[18] L. D. George Dahl, Dong Yu and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition."

[19] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning."

[20] A. Ridho, "Gender classification of indonesian names using multinomial naive bayes and random forest classifiers."

[21] S. Singh, "A pilot study on gender differences in conversational speech on lexical richness measures."

[22] J. F. S. Argamon, M. Koppel and A. Shimoni, "Gender, genre, and writing style in formal written texts."

[23] S. B. S. Herring, L. Scheidt and E. Wright, "Gender and genre variation in weblogs," p. 439– 459.

[24] C. A. T. Kucukyilmaz, B.B. Cambazoglu and F. Can, "Chat mining for gender prediction," p. 274–283.

[25] W. D. C. Peersman and L. Vaerenbergh, "Predicting age and gender in online social networks," pp. 37–44.

[26] K. G. R. Sarawgi and Y. Choi, "Gender attribution: Tracing stylometric evidence beyond topic and genre," pp. 78–86.

[27] S. A. M. Koppel and A. Shimoni, "Automatically categorizing written texts by author gender," p. 401–412.

[28] A. Mukherjee and B. Liu, "Improving gender classification of blog authors," p. 207–217.

[29] J. O. S. Nowson and A. Gill, "Gender, genres, and individual differences," p. 1666–1671.

[30] C. D. Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging."

[31] C. N. dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts."

[32] A. Karpathy, "Convolutional neural networks for visual recognition."

[33] D. D. Lewis, Y. Yang, T. G. Rose, and F. L. Rcv, "A new benchmark collection for text categorization research."

[34] V. Rus, P. M. M. M. C. Lintean, D. S. McNamara, and A. C. Graesser, "Paraphrase identification with lexico-syntactic graph subsumption."

[35] Keras website. [Online]. Available: https://keras.io