A Neural Network from Scratch

Shaishavkumar Jogani (1212392985), Siva Kongara (1212345483)

**Abstract**—These instructions give you guidelines for preparing papers for IEEE Computer Society Transactions. Use this document as a template if you are using Microsoft Word 6.0 or later. Otherwise, use this document as an instruction set. Please note that use of IEEE Computer Society templates is meant to assist authors in correctly formatting manuscripts for final submission and does not guarantee how the final paper will be formatted by IEEE Computer Society staff. This template may be used for initial submissions; however, please consult the author submission guidelines for formatting instructions as most journals prefer single column format for peer review. An abstract should be 100 to 200 words for regular papers, no more than 50 words for short papers and comments, and should clearly state the nature and significance of the paper. Abstracts *must not* include mathematical expressions or bibliographic references. Please note that abstracts are formatted as left justified in our editing template (as shown here).

—————————— ◆ ——————————

# 1 Introduction

Digit classification is the task of classifying a given image into one of the digits from 0 to 9. This is of key importance in modern day machine intelligence and has many practical applications such as Natural Language Processing (to autograde), Self Driving Cars (to detect speed limits) etc. This is a typical classification problem. A classification problem consists of a set of classes and the machine has to classify a given object into one of the classes. Other examples of a classification problem include facial recognition, image classification (identifying objects as car, bus, dog) etc. This classification problem can be addressed using many techniques such as Linear Regression, K-nearest-neighbours, SVM, Neural Network, Convolutional Nets and many other such machine learning techniques. In this project we have done the task of digit classification using a Neural Network. The motivation for this project comes up from the fact that digit classification has been an important research area for exploring several learning techniques ranging from automatically learning feature representations, learning classifiers invariant to distortions, matching and alignment based distances, and learning multilayered representations of data1.

# 2 Background

## 2.1 Neural Network

The term ‘Neural Network’ is inspired from the way biological nervous systems, such as the brain, process information. Human nervous system contains millions of neurons and each neuron propagates the information forward to another neuron. Based on the input signal that a neuron receives, it is either activated or not. The key idea of this paradigm is the novel structure of the artificial networks2. A Neural network (NN) is configured to recognize patterns or to classify the data. A NN consists of an input layer, any number of hidden layers (including 0), and an output layer. Each layer has a collection of neurons and all the layers are connected in an acyclic manner. In other words, the outputs of some neurons in a layer will be the inputs to other neurons in the above layer. Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including but not limited to sales forecasting, business marketing, customer research, data validation, risk management, medicine, robot learning etc.

The key difference between linear regression and a neural network is that the output can be modelled as a nonlinear function of the input. This nonlinearity arises because of the hidden layers. It is equivalent to mapping the input to higher dimensions and thus classifying the samples. In order to use a neural network for classification, we need to train the network with the available training examples and thus making the network learn the models for each of the classes and thus identifying a new test sample based on the learned models. Now, the important question is in what shape do these models get saved in the network and the answer to that question is the network parameters: weights and biases. Weights are the parameters that are used to weigh the incoming inputs to a neuron and thus suppressing some of the inputs while enhancing the others. Then the bias is added, and it is passed through an activation function which decides whether the neuron should be activated or not. After the bias is added to the weighted sum of a neuron’s input, the value could be anywhere between -inf to +inf. So, the activation function plays a key role here in deciding whether to fire that neuron based on the value. So, fundamentally training a network boils down to the problem of learning these weights such that the predicted result at the output layer is as close to the actual result. This difference is the error that the network has made in prediction and minimising this error should be our objective. There are many types of error functions that we can define when comparing the predicted result to the actual result. Some of them are the Mean Squared Error, Maximum Likelihood, Cross-Entropy. For the purpose of classification, cross-entropy loss function performs better than the other loss functions. This is because the problem of classification is a discrete one. The predicted value in this case is not a continuous numeric value, but a single class. So, if we have some kind of outputs for each class which represents the probabilities for the object belonging to that class, then we can use the Cross-Entropy loss to learn the parameters.

## 2.2 Stochastic Gradient Descent

Stochastic gradient descent (SGD) is a stochastic approximation of the gradient descent optimization and an iterative method for minimizing the loss function. Loss is defined as some kind of difference between the predicted output and the desired output. It is an iterative method in which the network is modified after each training sample is fed through the network and thus updating the network parameter in the direction in which the loss is minimised. SGD computes the gradient using a single sample and is much faster than the traditional batch descent algorithm. Large datasets often can't be held in RAM, which makes vectorization much less efficient. Rather, each sample or batch of samples must be loaded, worked with and the results are stored. This is computationally less expensive and also SGD is most useful when the objective function is non-convex. An objective function is termed non-convex if it has more than one local minima. In batch gradient descent we step down the true gradient and thus may eventually converge to local minima. SGD is better in finding the global minima. SGD involves two main steps: Forward Propagation and Backpropagation.

## 2.3 Forward Propagation

Fig. 2. Structure of our Neural Network

As discussed before, an artificial neural network (ANN) consists of an input layer, an output layer, and any number of hidden layers situated between the input and output layers. The feed-forward computations performed by the ANN are as follows: The signals from the input layer are multiplied by a set of fully-connected weights connecting the input layer to the hidden layer. These weighted signals are then summed and combined with a bias. This calculation forms the pre-activation signal for the hidden layer. The pre-activation signal is then transformed by the hidden layer activation function to form the feed-forward activation signals leaving leaving the hidden layer. Similarly, the activations of the other hidden layers are calculated from the previous hidden layer. In a similar fashion, the hidden layer activation signals are multiplied by the weights connecting the hidden layer to the output layer, a bias is added, and the resulting signal is transformed by the output activation function to form the network output. The output is then compared to a desired target and the error between the two is calculated. This entire process is called feed forward propagation.

The activation functions should be such that they should be differentiable, and this is important while propagating the error back through the network. Sigmoid activation is defined as follows: f(x) = 1/(1+e-x) and the derivative of the sigmoid activation is f’(x) = f(x)(1-f(x)). Also, a sigmoid function maps any arbitrarily large value to a range between 0 and 1. Thus, it can be thought as a function that decides whether a neuron is activated or not.

The softmax activation is also known as the normalized exponential function. This activation is generally applied to the output layer in a classification problem. It is defined as follows: f(zk) = eZk / Σ eZk for k = 1 to C where is the total number of classes to be classified into. This function reduces the value of each neuron in the output layer to be in the range of [0,1] and all the values of the output layer add up to 1. This behaviour is similar to probability and thus the outputs can be interpreted as the probability with which the given object belongs to each class. This can be represented as shown in the figure. Ideal output should be that these output values should have 1 for the class to which the object belongs and 0 for all the other classes.

## 2.3 Back Propagation

Backpropagation is the process of propagating the error that has been calculated at the end of forward propagation back through the network and update the network parameters so that the error is minimised. To do so, we make use of the chain rule to calculate the derivative of the loss with respect to parameters. Once we have the gradients of loss with respect to each of the network parameters, we update the parameter using a learning rate. This means that we change the values of the parameters in the direction in which the loss reduces.

We need to calculate the error between the predicted output and the actual output at the end of the forward propagation step. One type of such loss function is the Categorical Cross Entropy loss. The loss is defined as shown in the figure and the loss behaviour is shown in the figure..

In the above definition, *yi* is the predicted output (probability obtained after applying softmax activation) and *ti* is the actual output for the corresponding classes. Since this loss assumes that the predicted output is some kind of probability, it is often used along with the softmax activation on the output layer for a multi class classification problem. The main advantage of using cross entropy error over mean squared error is that the training doesn’t stall. This is because the predicted output should match either 0 or 1 for each output node and in mean squared error, the gradient that is propagated back contains a term (output)(1-output) which diminishes as the training progress. This leads to a very minimal change in the network parameters.

## 2.4 Dropout

Neural network with large number of parameters are very powerful but they are prone to the issue of overfitting. As the machine trains on the training samples, the machine tries to change its model (network parameters), to best fit the training data. This is not necessarily a desired thing as the main objective of the network is to classify/predict on a new test sample. Such a network which has 100% accuracy on the training data may not perform in general on a new test data. This is called the problem of overfitting as the network fits itself to the training data in a tight manner. This problem can be addressed by combining the predictions from many different networks at test time. But large networks are slow to train and use. So, Dropout is an effective way to prevent overfitting in the network. The key idea of Dropout is to randomly drop hidden units to prevent them from co-adapting too much. So, every time a sample is trained, some of the hidden units are dropped at random as shown in the figure. Such a network is called a thinned network and, so we train on many such thinned networks. These thinned networks are exponential in number and thus effectively giving us exponentially large number of models to predict the output from. At test time, we use all the units to predict the output and to have the output at test time to be equal to the expected output during training, we multiply the weights with the probability with which the unit is retained during the training.

# 3 IMPLEMENTATIONS

## 3.1 Architecture

In this project, we have implemented a neural network for the purpose of digit classification. Our implementation is in python and does not use any advanced libraries. There are two hidden layers with 256 neurons in each hidden layer and an output layer with 10 neurons corresponding to the ten digits from 0 to 9. The dataset that we have used in this project is the MNIST dataset in which each image is represented by 28 x 28 pixels. Each pixel’s value ranging from 0 to 255. We use these 784-pixel values to represent the input and to train the network. So, we have 784 neurons in the input layer. This constitutes the structure of our neural network.

## 3.2 Preparing the training data:

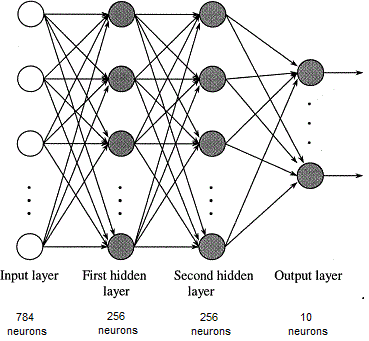


Fig. 1. Structure of our Neural Network

MNIST database of handwritten digits is used to train the network and to test the accuracy. The data is available in a byte format and we have used the python-mnist library to read the dataset which will return a list of all the images and the labels corresponding to each image. Each image returned by this library will be in the form of a list of 784 features corresponding to the 784 pixels with a value from 0 to 255. The label corresponding to that image is a number from 0 to 9. As the loss that we are about to calculate is categorical cross entropy, we have encoded the labels as one-hot vectors. So, that means a label of ‘2’ has been encoded as a list of 10 in which the second index is 1 and the rest are zeros i.e. [0,1,0,0,0,0,0,0,0,0]. This helps in comparison with the final output layer of the network in which only one neuron has to fire in an ideal situation.

There are 60000 images in the data set and we have sampled randomly 5000 out of them for the purpose of validation and 5000 for the purpose of testing. From the rest 50000, we have performed experiments changing the training size from 10000 to 50000. Since we have randomly sampled all the images, our assumption is that there are equal number of samples from each class in all the three splits. The image data has pixel values ranging from 0 to 255, so we have normalized each pixel value to be in the range [0,1] by dividing each value by 255. It makes the weak input weaker and the strong input stronger. This can be thought as each pixel either being on or off and it helps the algorithm to converge faster.

## 3.3 Network Parameters

We have defined six network parameters as follows:

*W1:* From input layer to the first hidden layer. Since input layer has 784 neurons and the first hidden layer has 256 neurons., this will be a 784 x 256 vector.

*B1:* This is the bias applied to each of the neuron in the first hidden layer, a 256 x 1 vector.

*W2:* From the first hidden layer to the second hidden layer. Since each of the hidden layers has 256 neurons, this will be a 256 x 256 vector.

*B2:* This is the bias applied to each of the neuron in the second hidden layer, a 256 x 1 vector.

*W3:* From the second hidden layer to the output layer. Since the second hidden layer has 256 neurons and the output layer has 10 neurons, this will be a 256 x 10 vector.

*B3:* This is the bias applied to each of the neuron in the output layer, a 10 x 1 vector.

Initially all the parameters are randomly initialised in the range of [0,1] with a mean of 0.5. However, we have observed that this initialisation lead to a very slow convergence and we have researched that initialising weights with a mean 0 would give us better results. So, we have initialised all the weights and biases in the range of [-1,1] with a mean of 0.

## 3.4 Stochastic Gradient Descent

We have implemented Stochastic Gradient Descent on each of the training sample in the training dataset. Each training sample is fed to the input and the output value is calculated through forward propagation. Then, using Back Propagation we have propagated the loss back to each layer and adjusted the weights and bias at each layer.

## 3.4 Forward Propagation

During forward propagation, at each of the hidden layers a sigmoid activation function is applied to each neuron in the hidden layers. For each of the output neuron, softmax activation is applied. The final output of the network is the result of this softmax activation and is used in estimating the loss and thus propagating the loss back through the network.

## 3.4 Dropout

To implement dropout in this network, we have defined a parameter called dropout rate and an element wise multiplication of each of the neuron in the hidden layer with a 256 x 1 mask generated out of a binomial distribution with dropout rate as its parameter. Our dropout rate parameter indicates the proportion of the number of hidden units retained.

## 3.4 Back Propagation

The error between the predicted output and the actual label which is encoded as a one hot vector is back propagated and the partial derivatives of the loss with respect to each of the network parameters are calculated. A learning rate of 0.1 is used initially, and it is decayed after every 5000 epochs at a rate of 0.005. Finally, the learning rate is fixed at 0.001.

After the partial derivatives of the loss with respect to the network parameters are determined, the parameters are updated.

Update step: H is the cross-entropy loss function and the update step is shown below for W1 and similar step is done for all the other parameters (W2, W3, B1, B2, B2).

ΔW1 = ẟH/ẟW1

W1 = W1 - η \* ΔW1

## 3.4 Validation and Testing

We After all the sample in the training dataset have been trained on and the parameters updated, the validation data is tested for accuracy and the next iteration of training is started based on the accuracy. In the next iteration, the training dataset is shuffled around so that each sample has no dependency on its previous or the next sample and is trained as if it is chosen independently. This leads to a faster convergence and is more efficient.

Once the training is completed, we have run the neural network on the test dataset to calculate the accuracy and thus measuring the performance of our digit classifier.

# 4 Results

The loss function of a neural network with one or more hidden layers with nonlinear activations is neither convex nor concave, but instead it is non-convex. The reason for this is the nonlinear activation functions. The nonlinear activation function corresponds to a non-convex optimization. So, a multilayered network with all linear activation function can still be convex but once a nonlinear activation function is used in any one of the layers, the optimization problem becomes non-convex because of the second order derivatives of the nonlinear activation functions are neither positive semidefinite, nor negative semidefinite. Also by swapping the parameters of nodes in a layer and doing the corresponding swaps in the layers above until the output layer, we can achieve the same loss. So, we have multiple solutions for the same loss value which are all the local minimas.

The purpose of a nonlinear activation function is to model a behavior which is not linear in nature. If all the layers have a linear activation function, then no matter how many layers we use, it would still behave as a single layer perceptron and is equivalent to a linear regression model which is not very powerful. Also, while back propagating the error, the gradient of a linear function is a constant and is not dependent on the input. So, we cannot determine the change of error with respect to a change in the input which is bad for learning the weights.

The way we update the weights is by back propagating the error back through the network. The objective is to calculate how the loss changes with respect to changes in the weights at each level in the network. We can calculate the partial derivative of the loss with respect any weight by traversing through the nodes from that weight to the output layer. In other words, this can be done in reverse order by first calculating the partial derivatives at the output layer and using the chain rule, we can calculate the derivatives at the layers below. So, by storing the results of these partial derivatives at the top layers, we can reuse them to calculate the derivatives at the layers below them. This is the core idea of dynamic programming: to reuse the results of a smaller problem to find the solution of a larger problem. This leads to an increase in the computational efficiency. So effectively backpropagation is a combination of using the chain rule and dynamic programming.

# 5 Equations

If you are using Word, use the MathType add-on (<http://www.mathtype.com>) for equations in your paper (Insert | Object | Create New | Microsoft Equation or MathType Equation). “Float over text” should not be selected.

For display equations as seen below, number equations consecutively with equation numbers in parentheses flush with the right margin, as in (1). First, use the equation editor to create the equation. Then, select the “Equation” markup style. Press the tab key and write the equation number in parentheses. To make your equations more compact, you may use the solidus ( / ), the exp function, or appropriate exponents. Use parentheses to avoid ambiguities in denominators. Punctuate equations when they are part of a sentence, as in

 (1)

Be sure that the symbols in your equation have been defined before the equation appears or immediately following. Italicize symbols (*T* might refer to temperature, but T is the unit tesla). Per IEEE Computer Society, please refer to “(1),” not “Eq. (1)” or “equation (1),” except at the beginning of a sentence: “Equation (1) shows ... .” Also see *The Handbook of Writing for the Mathematical Sciences*, 1993. Published by the Society for Industrial and Applied Mathematics, this handbook provides some helpful information about math typography and other stylistic matters. For further information about typesetting mathematical equations, please visit the IEEE Computer Society style guide: <http://www.computer.org/portal/web/publications/style_math>.

Please note that math equations might need to be reformatted from the original submission for page layout reasons. This includes the possibility that some in-line equations will be made display equations to create better flow in a paragraph. If display equations do not fit in the two-column format, they will also be reformatted. Authors are strongly encouraged to ensure that equations fit in the given column width.

# 6 Helpful Hints

## 6.1 Figures and Tables

Because IEEE Computer Society staff will do the final formatting of your paper, some figures may have to be moved from where they appeared in the original submission. Figures and tables should be sized as they are to appear in print. Figures or tables not correctly sized will be returned to the author for reformatting.

Detailed information about the creation and submission of images for articles can be found at <http://www.computer.org/portal/web/peerreviewjournals/author#figures> where you can [View transactions art\_guide.pdf (PDF, 4.69MB)](http://www.computer.org/cms/Computer.org/Journal%20templates/transactions_art_guide.pdf) . We strongly encourage authors to carefully review the material posted here to avoid problems with incorrect files or poorly formatted graphics.

Place figure captions below the figures; place table titles above the tables. Figure captions appear as left justified. Table captions are restricted to one sentence and are formatted as title case. Any additional sentence in a table caption will be formatted as a footnote below the table (see Table 1 in this document). If your figure has two parts, include the labels “(a)” and “(b)” as part of the artwork. Please verify that the figures and tables you mention in the text actually exist. Figures and tables should be called out in sequential order, as this is how they will be placed in your paper. For example, avoid referring to figure “8” in the first paragraph of the article unless figure 8 will again be referred to after the reference to figure 7. **Please do not include figure captions as part of the figure. Do not put captions in “text boxes” linked to the figures. Do not put borders around the outside of your figures.** Per IEEE Computer Society, please use the abbreviation “Fig.” even at the beginning of a sentence. Do not abbreviate “Table.” Tables are numbered numerically.

For journals that use print for publication, please verify with IEEE Computer Society that the journal you are submitting to does indeed accept color before submitting final materials. **Do not use color unless it is necessary for the proper interpretation of your figures.**

Figures (graphs, charts, drawing or tables) should be named fig1.eps, fig2.ps, etc. If your figure has multiple parts, please submit as a single figure. Please do not give them descriptive names. Author photograph files should be named after the author’s LAST name. Please avoid naming files with the author’s first name or an abbreviated version of either name to avoid confusion. If a graphic is to appear in print as black and white, it should be saved and submitted as a black and white file (grayscale or bitmap.) If a graphic is to appear in color, it should be submitted as an RGB color file.

Figure axis labels are often a source of confusion. Use words rather than symbols. As an example, write the quantity “Magnetization,” or “Magnetization *M*,” not just “*M*.” Put units in parentheses. Do not label axes only with units. As in Fig. 1, for example, write “Magnetization (A/m)” or “Magnetization (Am−1),” not just “A/m.” Do not label axes with a ratio of quantities and units. For example, write “Temperature (K),” not “Temperature/K.” Table 1 shows some examples of units of measure.

Multipliers can be especially confusing. Write “Magnetization (kA/m)” or “Magnetization (103 A/m).” Do not write “Magnetization (A/m) × 1,000” because the reader would not know whether the top axis label in Fig. 1 meant 16,000 A/m or 0.016 A/m. Figure labels should be legible, approximately 8 to 12 point type. When creating your graphics, especially in complex graphs and charts, please ensure that line weights are thick enough that when reproduced at print size, they will still be legible. We suggest at least 1 point.

## 6.3 Footnotes

Number footnotes separately in superscripts (Insert | Footnote)[[1]](#footnote-1). Place the actual footnote at the bottom of the column in which it is cited; do not put footnotes in the reference list (endnotes). Use letters for table footnotes (see Table 1). Please do not include footnotes in the abstract and avoid using a footnote in the first column of the article. This will cause it to appear above the affiliation box, making the layout look confusing.

TABLE 1  
Units for Magnetic Properties



Statements that serve as captions for the entire table do not need footnote letters.

aGaussian units are the same as cgs emu for magnetostatics; Mx = maxwell, G = gauss, Oe = oersted; Wb = weber, V = volt, s = second, T = tesla, m = meter, A = ampere, J = joule, kg = kilogram, H = henry.

## 6.4 Lists

The IEEE Computer Society style is to create displayed lists if the number of items in the list is longer than three. For example, within the text lists would appear 1) using a number, 2) followed by a close parenthesis. However, longer lists will be formatted so that:

1. Items will be set outside of the paragraphs.
2. Items will be punctuated as sentences where it is appropriate.
3. Items will be numbered, followed by a period.

## 6.5 Theorems and Proofs

Theorems and related structures, such as axioms corollaries, and lemmas, are formatted using a hanging indent paragraph. They begin with a title and are followed by the text, in italics.

**Theorem 1.** *Theorems, corollaries, lemmas, and related structures follow this format. They do not need to be numbered, but are generally numbered sequentially.*

Proofs are formatted using the same hanging indent format. However, they are not italicized.

**Proof.** The same format should be used for structures such as remarks, examples, and solutions (though these would not have a Q.E.D. box at the end as a proof does). 

# 7 End Sections

## Appendices

Appendices, if present, appear online as supplemental material. In the event multiple appendices are required, they will be labeled “Appendix A,” “Appendix B, “ etc.

IEEE Computer Society Transactions accepts supplemental materials for review with regular paper submissions. These materials may be published on our Digital Library with the electronic version of the paper and are available for free to Digital Library visitors. Please see our guidelines below for file specifications and information. Any submitted materials that do not follow these specifications will not be accepted. All materials must follow US copyright guidelines and may not include material previously copyrighted by another author, organization or company. More information can be found at <http://www.computer.org/portal/web/peerreviewjournals/author#supplemental>.

## 7.2 Acknowledgments

The preferred spelling of the word “acknowledgment” in American English is without an “e” after the “g.” Use the singular heading even if you have many acknowledgments. Avoid expressions such as “One of us (S.B.A.) would like to thank ... .” Instead, write “F. A. Author thanks ... .” Sponsor and financial support acknowledgments are included in the acknowledgment section. For example: This work was supported in part by the US Department of Commerce under Grant BS123456 (sponsor and financial support acknowledgment goes here). Researchers that contributed information or assistance to the article should also be acknowledged in this section. Also, if corresponding authorship is noted in your paper it will be placed in the acknowledgment section. Note that the acknowledgment section is placed at the end of the paper before the reference section.

## 7.3 References

Unfortunately, the Computer Society document translator cannot handle automatic endnotes in Word; therefore, type the reference list at the end of the paper using the “References” style. See the IEEE Computer Society’s style for reference formatting at: <http://www.computer.org/portal/web/publications/style_refs>. The order in which the references are submitted in the manuscript is the order they will appear in the final paper, i.e., references submitted nonalphabetized will remain that way.

Please note that the references at the end of this document are in the preferred referencing style. Within the text, use “et al.” when referencing a source with more than three authors. In the reference section, give all authors’ names; do not use “et al.” Do not place a space between an authors' initials. Papers that have not been published should be cited as “unpublished” [4]. Papers that have been submitted or accepted for publication should be cited as “submitted for publication” [5]. Please give affiliations and addresses for personal communications [6].

Capitalize all the words in a paper title. For papers published in journals not published in English, please give the English citation first, followed by the original foreign-language citation [7].

## 7.4 Additional Formatting and Style Resources

Additional information on formatting and style issues can be obtained in the IEEE Computer Society Style Guide, which is posted online at: <http://www.computer.org/portal/web/publications/styleguide>. Click on the appropriate topic under the Special Sections link.

# Conclusion

Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions. Authors are strongly encouraged not to reference multiple figures or tables in the conclusion—these should be referenced in the body of the paper.

**Acknowledgment**

The authors wish to thank A, B, C. This work was supported in part by a grant from XYZ.

**References**

1. Subhransu Maji and Jitendra Malik. “Fast and Accurate Digit Classification.” https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-159.pdf
2. Christos Stergiou and Dimitrios Siganos. “NEURAL NETWORKS.” https://www.doc.ic.ac.uk/~nd/surprise\_96/journal/vol4/cs11/report.html
3. Aleksander, I. and Morton, H. “An introduction to neural computing.” 2nd. edition
4. J.M.P. Martinez, R.B. Llavori, M.J.A. Cabo, and T.B. Pedersen, "Integrating Data Warehouses with Web Data: A Survey," *IEEE Trans. Knowledge and Data Eng.*, preprint, 21 Dec. 2007, doi:10.1109/TKDE.2007.190746.(PrePrint)

**First A. Author** Allbiographies should be limited to one paragraph consisting of the following: sequentially ordered list of degrees, including years achieved; sequentially ordered places of employ concluding with current employment; association with any official journals or conferences; major professional and/or academic achievements, i.e., best paper awards, research grants, etc.; any publication information (number of papers and titles of books published); current research interests; association with any professional associations. Author membership information, e.g., is a member of the IEEE and the IEEE Computer Society, if applicable, is noted at the end of the biography.

**Second B. Author Jr.** biography appears here.

**Third C. Author** biography appears here.

1. It is recommended that footnotes be avoided (except for the unnumbered footnote with the receipt date on the first page). Instead, try to integrate the footnote information into the text. [↑](#footnote-ref-1)