 doi:10.20944/preprints202208.0313.v2

Review

Convolutional Neural Network (CNN). A comprehensive overview

Anjeel Upreti*[1]

[1]Department of Information Management, Nepal Commerce Campus, Nepal
* Correspondence: practiceriwaz691@gmail.com

Abstract

Convolutional neural network (CNN), a class of artificial neural network (ANN) is attracting interests of researchers in all research domain. CNN was invented for computer vision. They have also shown to be useful for semantic parsing, sentence modeling and other natural language processing related tasks. Here in this paper we discuss the basics of CNN models and their scope to provide a reference/baseline to the researchers interested in using CNN models in their research.

Keywords: Convolutional Neural Network; domain; natural language processing; computer vision; semantic parsing

1. Introduction

Introduced by Lecun in 1989 [1], Convolutional Neural Network (CNN) is inspired by visual cortex. A CNN has input layers, multiple hidden layers and one output layer. Training starts by sending data to input layer. Then, the data are passed through a series of hidden layers. Hidden layer includes multiple filters, pooling layers. Appending these layers in between hidden layers produces better features [2]. Overview of CNN is provided in Figure 1. CNN was invented for computer vision [3]. They have also shown to be useful for semantic parsing, sentence modeling and other natural language processing (NLP) related tasks. Using a simple CNN model of one layer with pretrained word vectors has shown good performance boost improving state-of-art on 4 out of 7 benchmark datasets [4]. Movie Review (MR), Stanford Sentiment Treebank (SST-1), SST-2

 doi:10.20944/preprints202208.0313.v2

(same as SST-1 but with neutral reviews removed and binary labels), Subjectivity dataset (Subj), TREC question dataset (TREC), Customer reviews (CR) and Opinion polarity detection subtask of Multi-dimensional Personality Questionnaire (MPQA) dataset are the 7 datasets [5] experimented with. The four dataset that shows performance boost are MR, SST-2, CR and MPQA. He experimented with four different types of CNN model. These are the 4 CNN model [6] developed and experimented on. CNN-rand, CNN-static, CNN-non-static, CNN-multichannel where, CNN-rand is a model where the words are initialized randomly. The weights are updated during the training. In CNN-static, the weights from word2vec are used. Unknown words are updated during the training time. In CNN-non-static, all the weights of all the words are updated during the training. And in CNN- multichannel, two same word vectors are used, one is fine tuned and other is kept static. Use of dropout has proved to be a good regularizer improving performance by 2%–4%. They suggest that the pretrained vectors are good universal feature extractors and can be utilized across multiple dataset. He also made further observation on optimizers showing that Adadelta [7] produced similar results compared to Adagrad [8] with fewer epochs.

CNN also has been employed to extract relationships between two entities in a sentence [9]. Treating a sentence as sequential data and integrating word position information into CNN model has outperformed the state-of-art method in classifying the relation in a sentence. Their system includes three main components, Word Representation, Feature Extraction and Output. The words are converted to word vectors using word embedding technique. Lexical level features, Sentence level features (Word features, Position features) are combined together to form final feature vectors. These vectors are then fed to softmax classifier.

CNN also has been used for sentence modeling. The purpose of sentence modeling is to represent, classify, the semantic content of a sentence [10]. The network

 doi:10.20944/preprints202208.0313.v2

achieved high performance on question classification without using any hand engineered features. CNN extracted lexical and sentence level features. This shows that, without implementing parse tree, a network can be made applicable that can capture short and long-range relations in a sentence for any language.

By directly learning embedding or by applying CNN to high dimensional one hot vectors without using any external tools, sentiment classification and topic classification achieved state-of-art performances [11]. The network was trained without using bag-of-n-gram techniques or word vector techniques. There is also a paper describing the usage of CNN with different word embedding model such as sentence embedding model, mean word embedding model, or word embedding with bag of words [12]. By using CNN at sentence level they outperformed the natural language processing task by about 15%.

We can average word vector together from different datasets to generate or form a good representation of a sentence [13]. Also for text understanding CNN has been used [14]. They have demonstrated that, we can apply any level of inputs ranging from character level to abstract text concepts to CNN. This achieved a good performance without having any knowledge on words, phrases or sentences. Automatic detection of keywords, topic allocation are difficult language processing task. On a research by [15], they developed a model with long short term memory (LSTM) cells that can filter the unimportant words and detect salient keywords in the sentence. They have also outperformed the general sentence embedding method. Since LSTM can hold core knowledge for longer period of time, we can leverage it on sentence level.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 August 2022 doi:10.20944/preprints202208.0313.v2

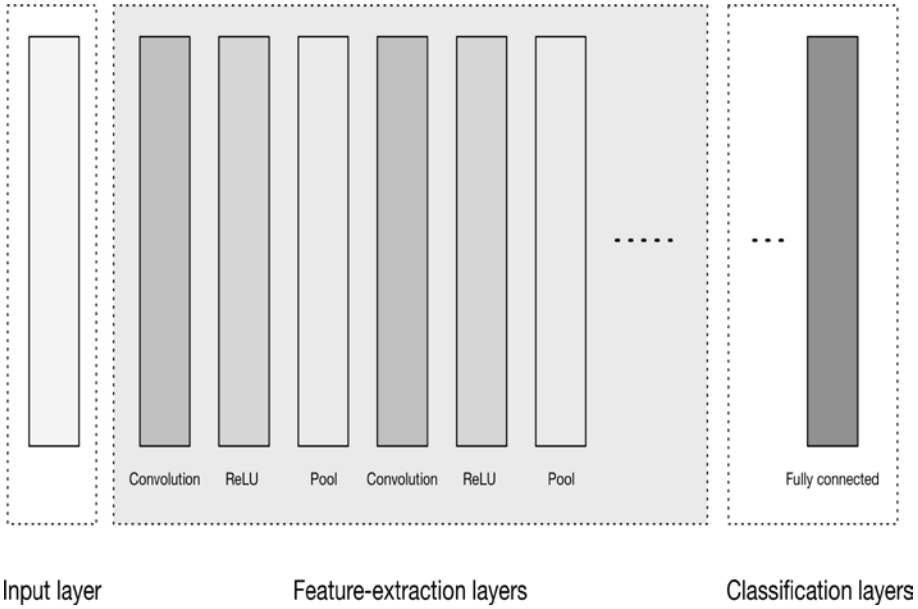


Figure 1: General Architecture of CNN (source: Google)

The components in Feature-extraction layers in Figure 1 are as:

- Convolution (see section 2.4.4)
- ReLU (see section 2.4.2)
- Pool (see section 2.4.5)

- **Applications of CNN**

  The main advantage of using CNN is that, CNN can itself learn features from the input removing the task of manual feature engineering. It is also more efficient in terms of memory and complexity than traditional fully connected neural network. Some of the applications of CNN are:

- Image Recognition
- Image Classification
- Speech Recognition

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 August 2022 doi:10.20944/preprints202208.0313.v2

- Text Classification

The use of CNN models have grown even higher recently, including in cloud based applications. As the technology is inclining towards cloud computing[16, 17], we assume that CNN will become more popular and its usage will grow more and more.

2. Neural network parameters

2.1. Shape

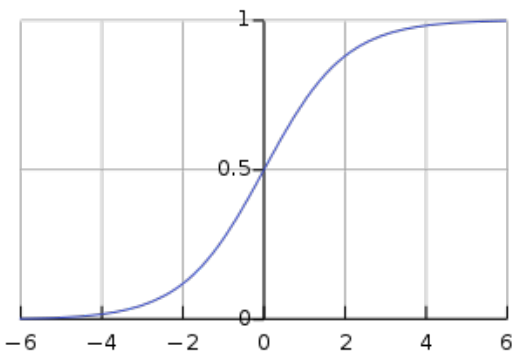
This is the size of the input data set. This shape can be of any dimension. For an image, shape (x,y) is representation of height and width of the image. For word embedding, shape (x,y) is representation is word and its vector size.

2.2. Activation function

Activation functions are used to increase the non-linearity of the network. This layer doesn't affect other layers. The main focus of activation layer is to determine whether the provided input should be activated or not [18].

2.3. Sigmoid

This activation is non linear in nature. The curve of this fuction is s-curve. The output of the fuction is always going to be in range of 0 and 1. When gradients are small, this activation functions cannot make significant changes. The sigmoid function is represented as below in Figure 2:



Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 August 2022 doi:10.20944/preprints202208.0313.v2

Figure 2: Characteristics of Sigmoid Activation Function (source: Google)

2.4. Rectified linear unit

This activation function is non linear in nature. This activation function has output 0 if the input is less than 0 and has output equal to input if the input is greater than 0. For large neural networks, Rectified Linear Unit (ReLU) can operate much faster than any other activation function. The ReLU function is represented as below in Figure 3.

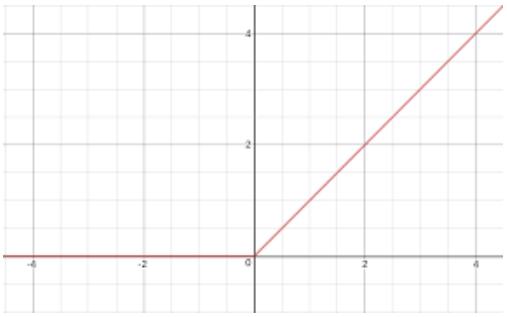


Figure 3: Characteristics of Rectified Linear Unit Activation Function (source:Google)

2.5. Hyperbolic tangent

tanH is a non-linear activation function. The range of tanh function is from -1 to 1. The shape of tanh is s - shaped. The main advantage of this activation function is that negative inputs will be mapped strongly negative. The Tanh function is represented as below in Figure 4.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 August 2022 doi:10.20944/preprints202208.0313.v2

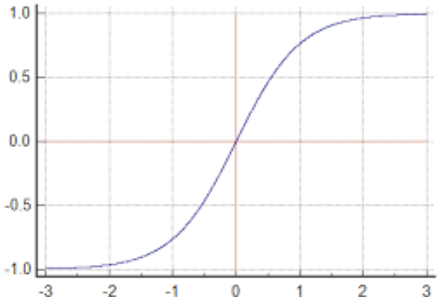


Figure 4: Characteristics of Hyperbolic Tangent Activation Function (source: Google)

2.6. Stride and padding

Stride is a metric for regulating the movement of convolutional filters for dot product operation in convolutional layer. It is the step for the convolution operation in convolutional layer. During feature mapping of convolution layer, operations on filters is performed by skipping steps either horizontally or vertically. This process is referred as stride. Padding is process of adding additional layer on the convolution layer. Data loss occurs when padding is not used [19]. For example, see Appendix A.1 for the operation of stride in convolutional layer.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 August 2022 doi:10.20944/preprints202208.0313.v2

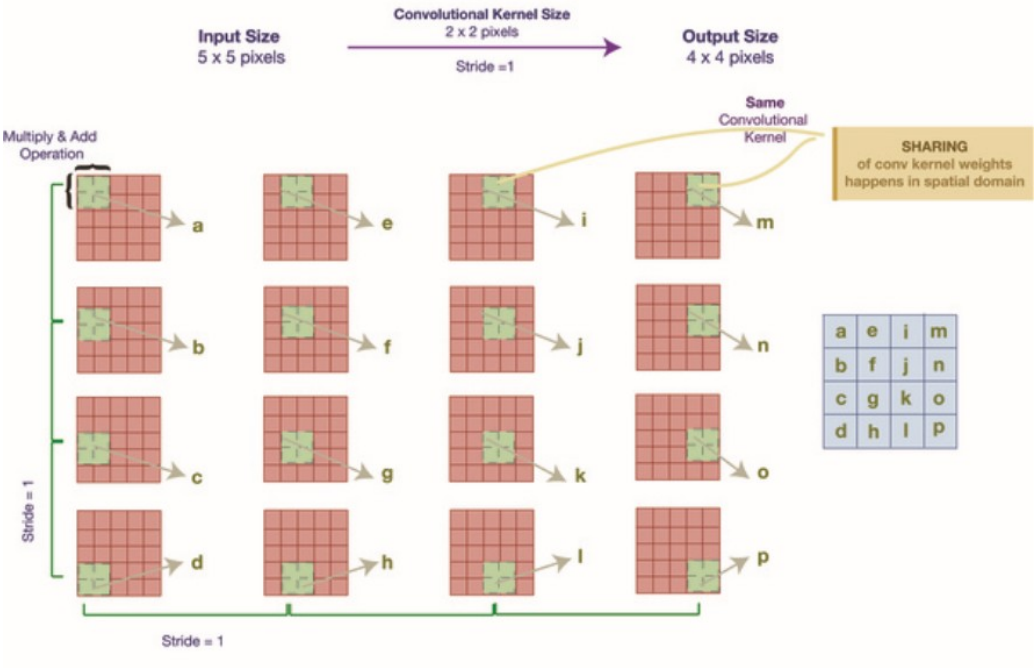


Figure 5: Stride operation when stride is 1 (source: Google)

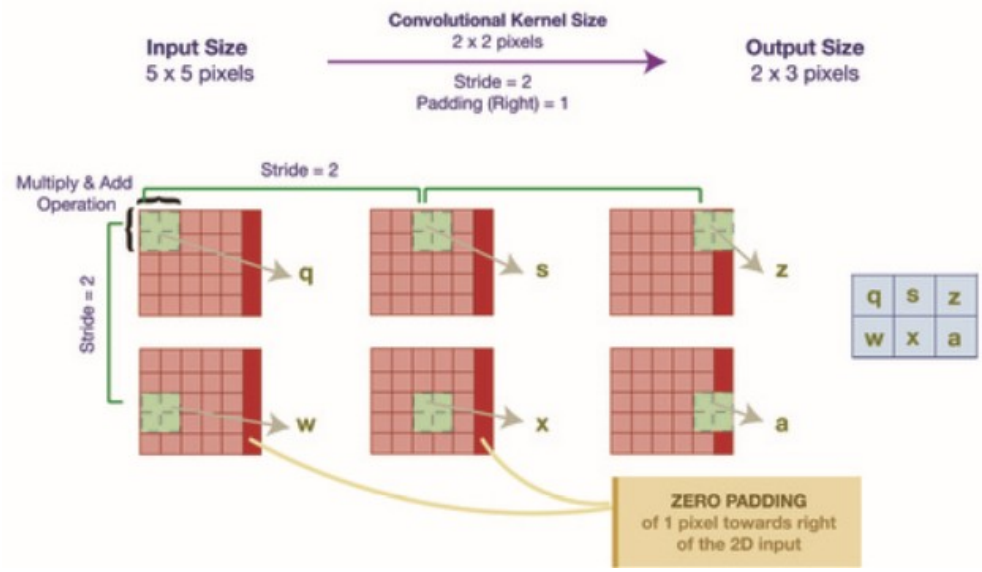


Figure 6: Stride operation when stride is 2 (source: Google)

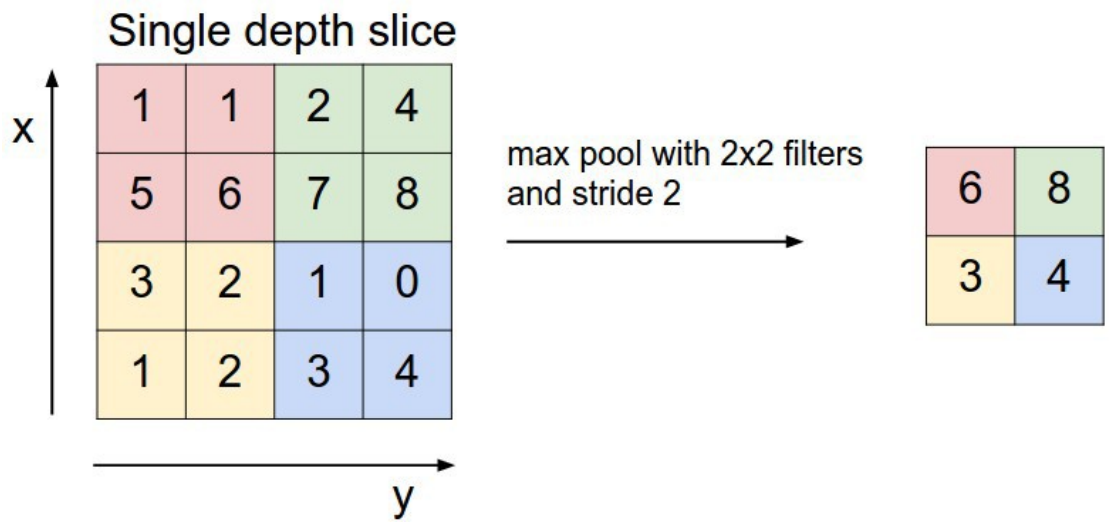
Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 August 2022 doi:10.20944/preprints202208.0313.v2

2.7. Convolutional layer and filter

Convolutional layer is the output of dot product between weights and inputs of previous layer. Neurons that covers the entire input and look for one feature are called filters. Many filters can be used on this convolutional layer [20]. The filter weights are shared across the receptive fields. These filter can help in determining features for the inputs [21]. These filters are 2 dimensional. Convolutional layers are spatially invariant. This means that they look for the same features across the entire input vectors. Input of this layer can be of multiple channels. For an example: an image has 3 channels RGB (Red, Green, Blue). These channels are fed to the layer for feature mapping operation.

2.8. Pooling layer

Pooling layer is the process of down sampling the convolutional layer. It depends upon the type of pooling layer for the selection of elements from convolutional layer during down sampling. For example: Max Pooling chooses the maximum number from a group of elements [22]. For example: see Appendix A.2 for the operation on max pooling.



 doi:10.20944/preprints202208.0313.v2

Figure 7: Max pooling sample with 2 x 2 filter (source: Google)

2.9. Dropout

Dropout is the process of ignoring the activation during the process of training. Specified percent of network's activation are ignored. However, it is not implemented during test phase. Usage of dropouts prevents overfitting by reducing the correlation between neurons [23].

2.10. Softmax

Softmax is used in classification problem. This function squashes the output of all unit between 0 and 1. The output of this function is equal to categorical probability distribution. i.e. this function can describe the probability for the class from output.

2.11. Batch normalization

Batch normalization is the process of initializing weights of the neural network. This reduces the initialization of bad weights for the networks. They are used just before activation layer [24].

2.12. Gradient descent

Gradient descent is the process of finding the minimum of a function. It uses learning rate to take steps from a initialized point. Depending upon the gradient at that new step, it is determined whether to take proportional forward steps or go backward [25].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 August 2022 doi:10.20944/preprints202208.0313.v2

2.13. Batch gradient descent

The main purpose of using batch gradient descent is to reduce the amount of data when computing the gradients for each learning step. It is the process when we sum up all on each iteration when performing the updates to the parameters. It is guaranteed to converge when batch gradient descent is used. Fix learning rate can be used when using this gradient descent without worrying about the learning rate decay. The batch size is the number of samples that is fed through the network.

When batch size is higher, more memory space is needed. For an example:
Assumptions:

Total datasets size=1000 Batch
size=100

then,

Total number of iteration taken =1000 / 100 =10 to complete 1 epoch where,

number of iteration is the number of passes and one pass is combination of one forward pass and one backward pass.

epoch is one forward pass and one backward pass of all data in datasets.

2.14. Training parameters

Neurons in convolutional layers has a local receptive field. This means that those neurons are not fully connected to the entire input but just some section of the input. Feature map is produced when we take input data together from abstraction provided by neurons. No any parameters has to be calculated in pooling layer and Dropout layer.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 August 2022 doi:10.20944/preprints202208.0313.v2

The below example describes about the parameter calculation for neural network.

Initials:

-a convolutional layer of shape $x, y, z(1, 28, 28)$

- where fi as input feature map (of value 1, from x=1)
- fo as output feature map (of value 32)
- filter size as $m \times n$ (of value 5, 5) then,

total number of weights=$n \times m \times fi \times fo$ (without bias) total number of weights= $(n \times m \times fi + 1) \times fo$ (with bias)

using above equation;we get :

total number of weights= $(5 \times 5 \times 1 + 1) \times 32 = 832$

2.15. Forget gate

This gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget and the closer to 1 means to keep.

2.16. Input gate

To update the cell state input gate is needed.Previous hidden states and current input are passed into

-a sigmoid function (SGF)

-a tanh function (THF)

Then multiply SGF and THF

 doi:10.20944/preprints202208.0313.v2

The sigmoid output will decide which information is important to keep from the tanH output.

2.17. Output gate

The output gate decides what the next hidden state should be. Hidden state contains information on previous inputs. The hidden input is also used for predictions. First the previous hidden state and the current input are passed into a sigmoid function. Then the newly modified cell state is passed to the tanH function. Then, the tanH outout is multiplied with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden state is then carried over to the next time step.

2.18. Cell state

Cell state from previous cell is (point wise) multiplied by forget vector. This has a probability of dropping values in the cell state if it gets multiplied by values near0. Then, point wise addition is done on the output from the input gate and current cell states that updates the cell state to new values that the neural network finds relevant. This gives us new cell state.

3. Conclusion

CNN is one of the most popular deep learning models. CNN comes handy for the complicated tasks like natural language processing, recommendation system, object detection, classification. Its dense network performs the task efficiently. We provided a comprehensive overview of this model structure in this paper which could be used as a touchstone by the researchers interested in using this model.

REFERENCES

[1] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *arXiv e-prints*, no. 1, p. arXiv:1408.5882, Aug. 2014. x, 15, 24

[2] M. Duan, E. Hill, and M. White, "Generating disambiguating paraphrases for structurally ambiguous sentences," in *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016, LAW@ACL 2016, August 11, 2016, Berlin, Germany*, 2016. [Online]. Available: http://aclweb.org/anthology/W/W16/W16-1718.pdf 1

[3] "Vector representations of words | tensorflow core | tensorflow," Tech. Rep. 3, 2019. [Online]. Available: https://www.tensorflow.org/tutorials/representation/word2vec 5

[4] J. Y. Lee and F. Dernoncourt, "Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks," *arXiv e-prints*, p. arXiv:1603.03827, Mar. 2016. 6, 17

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 2015, pp. 1–9. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7298594 6

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735 7

[7] F. A. Gers, J. Schmidhuber, and F. A. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, pp. 2451– 2471, 2000. [Online]. Available: https://doi.org/10.1162/089976600300015015

[8] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 28, pp. 2222–2232, 2017. [Online]. Available: https://doi.org/10.1109/TNNLS.2016.2582924 7

[9] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *CoRR*, vol. abs/1710.05941, 2017. [Online]. Available: http: //arxiv.org/abs/1710.05941

[10] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *CoRR*, vol. abs/1603.07285, 2016. [Online]. Available: http://arxiv.org/abs/1603.07285 11, 17

[11] Q. Huang, S. K. Zhou, S. You, and U. Neumann, "Learning to prune filters in convolutional neural networks," *CoRR*, vol. abs/1801.07365, 2018. [Online]. Available: http://arxiv.org/abs/1801.07365 11

[12] B. Athiwaratkun and K. Kang, "Feature representation in convolutional neural networks," *CoRR*, vol. abs/1507.02313, 2015. [Online]. Available: http://arxiv.org/abs/1507.02313 11

[13] D. Scherer, A. C. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks - ICANN 2010 -*

20th International Conference, Thessaloniki, Greece, September 15-18, 2010, Proceedings, Part III, 2010, pp. 92–101.
[Online]. Available: https://doi.org/10.1007/978-3-642-15825-4_10 12, 17, 62

[14] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdi- nov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online].
Available: http://dl.acm.org/citation.cfm?id=2670313 12, 28

[15] Prasai, R. (2022a). Earth engine application to retrieve long-term terrestrial and aquatic time series of satellite reflectance data. International Journal of Multidisciplinary Research and Growth Evaluation, 165–171. https://doi.org/10.54660/anfo.2022.3.3.11

[16] Prasai, R. (2022b). An open-source web-based tool to perform spatial multicriteria analysis. International Journal of Multidisciplinary Research and Growth Evaluation, 297–301. https://doi.org/10.54660/anfo.2022.3.3.19

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 448–456. [Online]. Available:
http://jmlr.org/proceedings/papers/v37/ioffe15.html

[18] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609. 04747 12, 17

[19] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L.

 doi:10.20944/preprints202208.0313.v2

D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural Computation*, vol. 1, pp. 541–551, 1989. [Online].
Available: https://doi.org/10.1162/neco.1989.1.4.541 15

[20] M. D. Zeiler, “ADADELTA: an adaptive learning rate method,” *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701 16

[21] J. C. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011. [Online]. Available:
http://dl.acm.org/citation.cfm?id=2021068 16

[22] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, “Relation classification via convolutional deep neural network,” in *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, 2014, pp. 2335–2344. [Online]. Available: http://aclweb.org/anthology/C/C14/C14-1220.pdf 16

[23] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A convolutional neural network for modelling sentences,” *CoRR*, vol. abs/1404.2188, 2014. [Online].
Available: http://arxiv.org/abs/1404.2188 16

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781 16, 17

[25] X. Zhang and Y. LeCun, “Text understanding from scratch,” *CoRR*, vol. abs/1502.01710, 2015. [Online]. Available: http://arxiv.org/abs/1502.01710 16

 doi:10.20944/preprints202208.0313.v2