An Introduction to Convolutional Neural Networks(CNN)

***Abstract*—The field of machine learning has taken a dramatic twist in re- cent times, with the rise of the Artificial Neural Network (ANN). These biologically inspired computational models are able to far exceed the per- formance of previous forms of artificial intelligence in common machine learning tasks. One of the most impressive forms of ANN architecture is that of the Convolutional Neural Network (CNN). CNNs are primarily used to solve difficult image-driven pattern recognition tasks and with their precise yet simple architecture, offers a simplified method of getting started with ANNs. This article offers a succinct introduction to CNNs, addressing freshly released papers and newly developed methods in creating these brilliantly spectacular image recognition models. This introduction will presume that you are already acquainted with the basics of ANNs and machine learning.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Artificial Neural Networks (ANNs) are computer processing systems that draw a lot of their inspiration from how organic nerve systems, like the human brain, function. ANNs are primarily made up of a large number of connected computational nodes (also known as neurons), which collaborate in a distributed manner to learn from the input and optimize the output. A model of an ANN's fundamental composition may be seen in Figure 1. The input would be loaded into the input layer, which would then distribute it to the hidden layers. The input is typically in the form of a multidimensional vector. The learning process is when the hidden layers consider judgments from the preceding layer and determine if a stochastic change inside itself worsens or improves the output. Deep learning is a term used to describe the stacking of many hidden layers.

In tasks involving image processing, supervised and unsupervised learning are the two main learning paradigms. Learning with pre-labeled inputs that serve as objectives is known as supervised learning. There will be a set of input values (vectors) and one or more associated defined output values for each training example. This kind of training aims to lower the overall classification error of the model by accurately calculating the training example-by-training output value. Unsupervised learning differs from supervised learning in that there are no la- bels in the training set. The ability of the network to decrease or enhance an associated cost function often serves as a metric for success. It is crucial to remember that the majority of challenges for pattern recognition that concentrate on images often rely on categorization utilizing supervised learning. Similar to conventional ANNs, convolutional neural networks (CNNs) are made up of neurons that

Identify applicable funding agency here. If none, delete this.

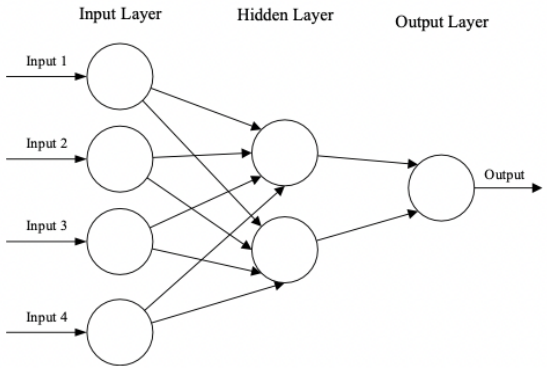

Fig. 1. : A simple three layered feedforward neural network (FNN), comprised of a input layer, a hidden layer and an output layer. This structure is the basis of a number of common ANN architectures, included but not limited to Feedforward Neural Networks (FNN), Restricted Boltzmann Machines (RBMs) and Recurrent Neural Networks (RNNs).

learn to optimize themselves. The fundamental building block of innumerable ANNs, each neuron will continue to take in input and carry out an action (such as a scalar product followed by a non-linear function). The whole network will still express a single perceptual scoring function from the input raw picture vectors to the class score at the end (the weight). The last layer will include loss functions related to the classes, and all of the standard techniques used for conventional ANNs are still applicable. The fact that CNNs are largely used in the area of pattern detection inside pictures is the sole significant distinction between CNNs and conventional ANNs. This enables us to include image-specific elements into the design, improving the network's suitability for image-focused tasks and lowering the number of setup parameters. Traditional ANN models often struggle with the computational complexity needed to calculate picture data, which is one of their biggest drawbacks. Due to its comparatively low picture dimensionality of only 28 28, common machine learning benchmarking datasets like the MNIST database of handwritten digits are appropriate for most types of ANN. Because MNIST is normalized to just black and white values, a single neuron in the first hidden layer will have 784 weights (28 28 1), which is reasonable for most ANN types. The number of weights on only one neuron of the first layer significantly rises to 12, 288 when you take into account a larger colored picture input of 64 64. You can see the disadvantages of adopting such models by keeping in mind that the network will need to be much bigger than one used to identify colour-normalized MNIST digits in order to

handle this scale of input.

A. Overfitting

Why, therefore, does it matter? It must be possible to simply expand the number of hidden layers in our network and maybe the number of neurons contained inside them. No is the obvious response to this question. The first is the obvious issue of not having infinite processing power and time to train these enormous ANNs. The second justification is to halt or lessen the consequences of overfitting. When a network is unable to learn efficiently for a variety of reasons, it is said to be overfit. Every effort should be made to minimize its impacts since it is a key notion in the majority, if not all, machine learning algorithms. If our models showed evidence of overfitting, we could be less able to identify generalized characteristics in our training dataset, as well as in our test and prediction sets. Our ANNs' complexity has been reduced primarily for this reason. The network will be less prone to overfit if fewer parameters are needed to train it, which will also increase the model's ability to predict the future.

II. CNN ARCHITECTURE

As was already said, CNN bases its coverage mainly on the assumption that the input would be composed of pictures. This concentrates the architecture's setup to best meet the requirements for handling the particular type of data. One of the most important differences is that the layers of the CNN are made up of neurons organized into three dimensions: the depth, the width, and the height of the input space. The depth describes the third dimension of an activation volume rather than the total number of layers inside the ANN. In contrast to conventional ANNS, each layer's neurons will only link to a tiny portion of the layer before it. Since we would have condensed the entire input dimensionality into a smaller volume of class scores filed across the depth dimension, in practice this would mean that for the example given earlier, the input "vol- ume" will have a dimensionality of 64 64 3 (height, width, and depth). This would then result in a final output layer composed of a dimensionality of 1 1 n (where n represents the possible number of classes).

A. Overall architecture

CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully-connected layers. When these layers are stacked, a CNN architecture has been formed. A simplified CNN architecture for MNIST classification is illustrated in Figure 2.

The basic functionality of the example CNN above can be broken down into four key areas. 1. As found in other forms of ANN, the input layer will hold the pixel values of the image. 2. The convolutional layer will determine the output of neurons of which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input vol- ume. The rectified linear unit (commonly shortened to ReLu) aims to apply an 'elementwise' activation function such as sigmoid to the output

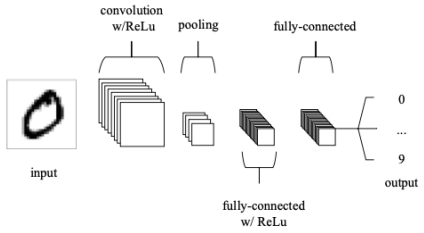

Fig. 2. An simple CNN architecture, comprised of just five layers

of the activation produced by the previous layer. 3. The pooling layer will then simply perform downsampling along the spatial dimensionality of the given input, further reducing the number of pa- rameters within that activation. 4. The fully-connected layers will then perform the same duties found in standard ANNs and attempt to produce class scores from the activations, to be used for classification. It is also suggested that ReLu may be used between these layers, as to improve performance.

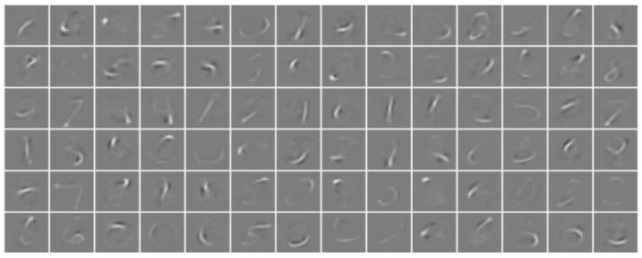

Fig. 3. Activations taken from the first convolutional layer of a simplistic deep CNN, after training on the MNIST database of handwritten digits. If you look carefully, you can see that the network has successfully picked up on characteristics unique to specific numeric digits.

Through this simple method of transformation, CNNs are able to transform the original input layer by layer using convolutional and downsampling tech- niques to produce class scores for classification and regression purposes. However, it is important to note that simply understanding the overall archi- tecture of a CNN architecture will not suffice. The creation and optimisation of these models can take quite some time, and can be quite confusing. We will now explore in detail the individual layers, detailing their hyperparameters and connectivities.

B. Convolutional layer

As the name implies, the convolutional layer plays a vital role in how CNNs operate. The layers parameters focus around the use of learnable kernels.These kernels are usually small in spatial dimensionality, but spreads along the entirety of the depth of the input. When the data hits a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input to produce a 2D activation map. These activation maps can be visualised, as seen in Figure 3.

As we glide through the input, the scalar product is calculated for each value in that kernel. (Figure 4) From this the network will learn kernels that 'fire' when they see a specific feature at a given spatial position of the input. These are commonly known as activations.Fig. 4:

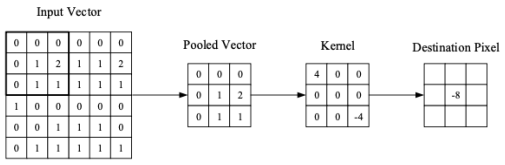

Fig. 4. : A visual representation of a convolutional layer. The centre element of the kernel is placed over the input vector, of which is then calculated and replaced with a weighted sum of itself and any nearby pixels.

A visual representation of a convolutional layer. The centre element of the kernel is placed over the input vector, of which is then calculated and replaced with a weighted sum of itself and any nearby pixels. Every kernel will have a corresponding activation map, of which will be stacked along the depth dimension to form the full output volume from the convolutional layer. As we alluded to earlier, training ANNs on inputs such as images results in models of which are too big to train effectively. This comes down to the fully- connected manner of standard ANN neurons, so to mitigate against this every neuron in a convolutional layer is only connected to small region of the input volume. The dimensionality of this region is commonly referred to as the re- ceptive field size of the neuron. The magnitude of the connectivity through the depth is nearly always equal to the depth of the input. For example, if the input to the network is an image of size 64 × 64 × 3 (a RGB- coloured image with a dimensionality of 64 × 64) and we set the receptive field size as 6 × 6, we would have a total of 108 weights on each neuron within the convolutional layer. (6 × 6 × 3 where 3 is the magnitude of connectivity across the depth of the volume) To put this into perspective, a standard neuron seen in other forms of ANN would contain 12, 288 weights each. Convolutional layers are also able to significantly reduce the complexity of the model through the optimisation of its output. These are optimised through three hyperparameters, the depth, the stride and setting zero-padding.The depth of the output volume produced by the convolutional layers can be manually set through the number of neurons within the layer to a the same region of the input. This can be seen with other forms of ANNs, where the all of the neurons in the hidden layer are directly connected to every single neuron beforehand. Reducing this hyperparameter can significantly minimise the total number of neurons of the network, but it can also significantly reduce the pattern recognition capabilities of the model. We are also able to define the stride in which we set the depth around the spatial dimensionality of the input in order to place the receptive field. For example if we were to set a stride as 1, then we would have a heavily overlapped receptive field producing extremely large activations. Alternatively, setting the stride to a greater number will reduce the amount of overlapping and produce an output of lower spatial dimensions.

C. Pooling layer

Pooling layers aim to gradually reduce the dimensionality of the representa- tion, and thus further reduce the number of parameters and the computational complexity of the model. The pooling layer operates over each activation map in the input, and scales its dimensionality using the "MAX" function. In most CNNs, these come in the form of max-pooling layers with kernels of a dimensionality of 2 × 2 applied with a stride of 2 along the spatial dimensions of the input. This scales the activation map down to 25Due to the destructive nature of the pooling layer, there are only two generally observed methods of max-pooling. Usually, the stride and filters of the pooling layers are both set to 2 × 2, which will allow the layer to extend through the entirety of the spatial dimensionality of the input. Furthermore overlapping pooling may be utilised, where the stride is set to 2 with a kernel size set to 3. Due to the destructive nature of pooling, having a kernel size above 3 will usually greatly decrease the performance of the model. It is also important to understand that beyond max-pooling, CNN architectures may contain general-pooling. General pooling layers are comprised of pooling neurons that are able to perform a multitude of common operations including L1/L2-normalisation, and average pooling. However, this tutorial will primar- ily focus on the use of max-pooling.

D. Fully-connected layer

The fully-connected layer contains neurons of which are directly connected to the neurons in the two adjacent layers, without being connected to any layers within them. This is analogous to way that neurons are arranged in traditional forms of ANN. (Figure 1)

III. Recipes

Despite the relatively small number of layers required to form a CNN, there is no set way of formulating a CNN architecture. That being said, it would be idiotic to simply throw a few of layers together and expect it to work. Through reading of related literature it is obvious that much like other forms of ANNs, CNNs tend to follow a common architecture. This common architecture is illus- trated in Figure 2, where convolutional layers are stacked, followed by pooling layers in a repeated manner before feeding forward to fully-connected layers.Another common CNN architecture is to stack two convolutional layers before each pooling layer, as illustrated in Figure 5. This is strongly recommended as stacking multiple convolutional layers allows for more complex features of the input vector to be selected.Fig. 5: A common form of CNN architecture in which convolutional layers are stacked between ReLus continuously before being passed through the pooling layer, before going between one or many fully connected ReLus.

It is also advised to split large convolutional layers up into many smaller sized convolutional layers. This is to reduce the

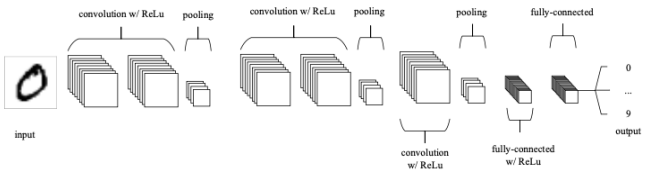

Fig. 5. : A common form of CNN architecture in which convolutional layers are stacked between ReLus continuously before being passed through the pooling layer, before going between one or many fully connected ReLus.

amount of computational complexity within a given convolutional layer. For example, if you were to stack three convolutional layers on top of each other with a receptive field of 3×3. Each neuron of the first convolutional layer will have a 3 × 3 view of the input vector. A neu- ron on the second convolutional layer will then have a 5 × 5 view of the input vector. A neuron on the third convolutional layer will then have a 7 × 7 view of the input vector. As these stacks feature non-linearities which in turn allows us to express stronger features of the input with fewer parameters. However, it is important to understand that this does come with a distinct memory allocation problem - especially when making use of the backpropagation algorithm. The input layer should be recursively divisible by two. Common numbers in- clude 32 × 32, 64 × 64, 96 × 96, 128 × 128 and 224 × 224. Whilst using small filters, set stride to one and make use of zero-padding as to ensure that the convolutional layers do not reconfigure any of the dimension- ality of the input. The amount of zero-padding to be used should be calculated by taking one away from the receptive field size and dividing by two.activation CNNs are extremely powerful machine learning algorithms, however they can be horrendously resource-heavy. An example of this problem could be in filter- ing a large image (anything over 128 × 128 could be considered large), so if the input is 227 × 227 (as seen with ImageNet) and we're filtering with 64 kernels each with a zero padding of then the result will be three activation vectors of size 227 × 227 × 64 - which calculates to roughly 10 million activations - or an enormous 70 megabytes of memory per image. In this case you have two op- tions. Firstly, you can reduce the spatial dimensionality of the input images by resizing the raw images to something a little less heavy. Alternatively, you can go against everything we stated earlier in this document and opt for larger filter sizes with a larger stride (2, as opposed to 1). In addition to the few rules-of-thumb outlined above, it is also important to ac- knowledge a few 'tricks' about generalised ANN training techniques. The au- thors suggest a read of Geoffrey Hinton's excellent "Practical Guide to Training Restricted Boltzmann Machines".

IV. Conclusion

Convolutional Neural Networks differ to other forms of Artifical Neural Net- work in that instead of focusing on the entirety of the problem domain, knowl- edge about the specific type of input is exploited. This in turn allows for a much simpler network architecture to be set up. This paper has outlined the basic concepts of Convolutional Neural Networks, explaining the layers required to build one and detailing how best to structure the network in most image analysis tasks. Research in the field of image analysis using neural networks has somewhat slowed in recent times. This is partly due to the incorrect belief surrounding the level of complexity and knowledge required to begin modelling these superbly powerful machine learning algorithms. The authors hope that this paper has in some way reduced this confusion, and made the field more accessible to beginners.

References

1. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 3642–3649. IEEE (2012)

2. Cires¸an, D.C., Giusti, A., Gambardella, L.M., Schmidhuber, J.: Mitosis detection in breast cancer histology images with deep neural networks. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013, pp. 411–418. Springer (2013)

3. Ciresan, D.C., Meier, U., Masci, J., Maria Gambardella, L., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. vol. 22, p. 1237 (2011)

4. Cires¸an, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Convolutional neural network committees for handwritten character classification. In: Document Analysis and Recognition (ICDAR), 2011 International Conference on. pp. 1135–1139. IEEE (2011)

5. Egmont-Petersen, M., de Ridder, D., Handels, H.: Image processing with neural networksa review. Pattern recognition 35(10), 2279–2301 (2002)

6. Farabet, C., Martini, B., Akselrod, P., Talay, S., LeCun, Y., Culurciello, E.: Hardware accelerated convolutional neural networks for synthetic vision systems. In: Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on. pp. 257–260. IEEE (2010)

7. Hinton, G.: A practical guide to training restricted boltzmann machines. Momentum 9(1), 926 (2010) 8. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)

9. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on 35(1), 221–231 (2013)

10. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Largescale video classification with convolutional neural networks. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. pp. 1725–1732. IEEE (2014)

11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In:

Advances in neural information processing systems. pp. 1097–1105 (2012)

12. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation 1(4), 541–551 (1989)

13. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)

14. Nebauer, C.: Evaluation of convolutional neural networks for visual recognition. Neural Networks, IEEE Transactions on 9(4), 685–696 (1998)

15. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: null. p. 958. IEEE (2003)

16. Srivastava, N.: Improving neural networks with dropout. Ph.D. thesis, University of Toronto (2013)

17. Szarvas, M., Yoshizawa, A., Yamamoto, M., Ogata, J.: Pedestrian detection with convolutional neural networks. In: Intelligent Vehicles Symposium, 2005. Proceedings. IEEE. pp. 224–229. IEEE (2005)

18. Szegedy, C., Toshev, A., Erhan, D.: Deep neural networks for object detection. In: Advances in Neural Information Processing Systems. pp. 2553–2561 (2013)

19. Tivive, F.H.C., Bouzerdoum, A.: A new class of convolutional neural networks (siconnets) and their application of face detection. In: Neural Networks, 2003. Proceedings of the International Joint Conference on. vol. 3, pp. 2157–2162. IEEE (2003)

20. Zeiler, M.D., Fergus, R.: Stochastic pooling for regularization of deep convolutional neural networks. arXiv preprint arXiv:1301.3557 (2013)

21. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Computer Vision–ECCV 2014, pp. 818–833. Springer (2014)