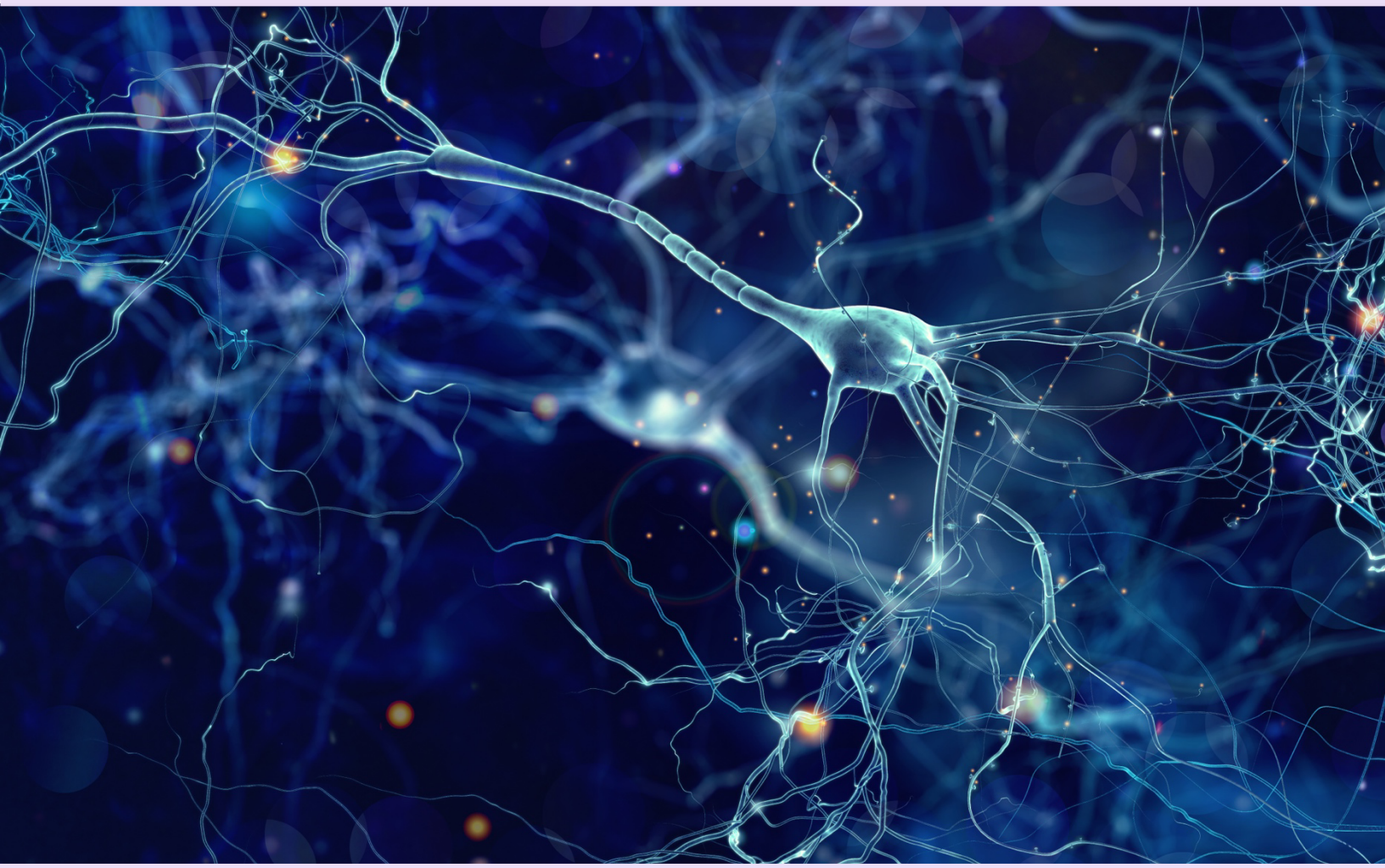
INTRODUCTION TO MODERN **CONVOLUTIONAL NETWORKS**



Centrella, Christopher
Franciscan University of Steubenville

INTRODUCTION

In recent years, artificial intelligence has emerged as one of the fastest-growing technologies in the computer industry. Everything from cars to computer software, to retail store shopping sites, all use artificial intelligence or AI in their product. In our ever-dynamic computer-driven society, artificial intelligence is used everywhere, in every way, and is at the forefront of our most modern cutting-edge technologies. But what exactly is artificial intelligence, how does it differ from traditional computing, and what are some of the underlying artificial intelligence strategies used today? In this paper, I wish to investigate this topic in greater depth, first introducing artificial intelligence and then expounding upon this to discuss what is at the heart of modern computer image recognition, convolutional neural networks.

WHAT IS ARTIFICIAL INTELLIGENCE?

First, what is AI? What does it mean to be "artificially intelligent"? Some people, with absolutely no knowledge of the subject, attribute artificial intelligence to playing the role of God in creating new living objects with arms and legs. Nothing could be further from the truth. Artificial intelligence does not involve any organic matter or living organism. Rather, artificial intelligence, at best is only a *simulation* of human intelligence.[1] Another way to describe it would be to say that artificial intelligence is the best attempt computers can make at imitating human behavior. This begs the question, in what ways do computers imitate human behavior? In *Artificial Intelligence for Dummies*, the authors describe four ways in which artificial intelligence attempts to mimic human behavior: Acting humanly, thinking humanly, thinking rationally, and acting rationally.[2]

Now there is an important distinction between acting humanly and acting rationally. John Paul Mueller and Luca Massaron, in the aforementioned book, explain

[1] (Mueller and Massaron 2022, Ch. 1)

[2] Ibid.

that acting rationally means that the system "always does the right thing based on the current information, given an ideal performance measure." In contrast, acting humanly involves other variables that can influence a person's behavior: instinct, intuition, and the like.[3] While any computer program acts rationally based on the person who programmed it, following a logic-based algorithm, or set of instructions,[4] using artificial intelligence allows the computer to act in a human way, which is much more than simply following a set of instructions. It includes simulating reason and intuition, even in very complex scenarios. Whereas with standard computer programming, every use case is handled explicitly by the programmer, with artificial intelligence the computer itself is making decisions.

There are several ways that artificial intelligence is able to mimic human behavior, but the primary way is through what is known as machine learning.[5] In machine learning, the computer abstracts functions from the data[6] to complete what is known as a model.[7] The skeleton of this model is created by the programmer, but the model is not finished until the individual *weights* are trained automatically, by the computer. The model represents a function, or a "deterministic mapping from a set of input values to one or more output values."[8] Each unique situation is represented by an input value, and the result is described through an output value. After the model is complete, the computer *infers* from it to apply the model to the situation at hand.[9] This allows the computer to quickly apply the model to new situations and new circumstances without having to retrain, thus allowing for the speed and performance of the iPhone 13, or the Tesla Model 3.

[3] (Mueller and Massaron 2022, Ch. 1)

[4] (Kelleher 2019, 7)

[5] (Artascanchez and Joshi 2020, Ch. 1)

[6] (Kelleher 2019, 8)

[7] Ibid, 12-14.

[8] Ibid, 7.

[9] Ibid, 14.

NEURAL NETWORKS

The scenario I described above represents a typical *neural network*, often referred to as a deep neural net. A neural network can be described as a mathematical model,[10] inspired by the operation of the human brain (see figure 1).[11] So the human brain consists of billions of tiny cells, called neurons.[12] Each of these cells gets data from other cells via *dendrites* and then outputs data to additional cells via a long rod, known as the *axon*.[13] See figure 1. A neural network has all of these characteristics. First, there are thousands of little computer "cells" which we refer to as *neurons*.[14] In neural networks, these neurons are grouped into layers,[15] where each output cell of a given layer, is connected to each input cell of the next layer. The layers consist of an input layer, "hidden" or intermediate layers, and one or more final output layers. In fact, the number of hidden or intermediate layers, and the output layer(s) is known as the *depth* of the network.[16] Now, each neuron has a unique *weight*; and the value of the neuron's input is multiplied by the value of the weight, to calculate that neuron's output.[17]

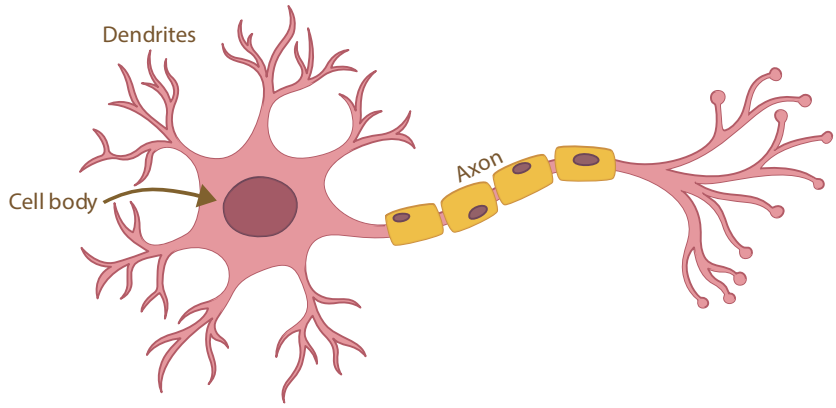


Figure 1. Brain Neuron
Credit: Adobe Stock

[10] (Kelleher 2019, 39)

[11] Ibid.

[12] Ibid, 65.

[13] Ibid, 66.

[14] Ibid, 67.

[15] Ibid, 67-8.

[16] Ibid, 68.

[17] Ibid, 70.

However, one last thing is needed, and that is the nonlinear activation function. So far, we have a bunch of layers, with a bunch of neurons; each layer takes in all the neurons from the previous layer, and then multiplies them by the appropriate given neuron values. The only issue is that this is a *linear* function.[18] A linear function means that if there is an increase or decrease in one thing, there is an increase or decrease in another thing at the same rate.[19] The problem is that in the real world, many things are not linear. If we limit an artificial intelligence model to a linear function, our model will automatically introduce errors, since it is constrained to only using linear functions, when our data uses a nonlinear function. To solve this problem, the concept of an *activation function* comes in.[20] An activation function applies a nonlinear mathematic function to the calculated value of the neuron, allowing the neuron's value to be nonlinear with the weight of the neuron, before it is input to the next layer.[21] Several activation functions exist, including the Sigmoid function, the SoftMax function, and the Rectified Linear Unit (ReLU) function.[22] The ReLU function is one of the best activation functions, because it reduces the problem of vanishing gradients, discussed later under the section "Backpropagation."[23]

CALCULATING LOSS

After we choose our activation function and the number of neurons that we want our model to have, the model has to choose the right values for each of the neurons, known as *weights*[24], or *parameters*.[25] These values are automatically generated by the

[18] (Kelleher 2019, 77-8)

[19] Ibid, 78.

[20] Ibid, 71, 76-9.

[21] Ibid.

[22] (Elgendy 2020, Ch. 2)

[23] Ibid.

[24] (Kelleher 2019, 70)

[25] (Elgendy 2020, Ch. 4)

neural network, and are the main way the system "learns."[26] Now, how does the system know which are the correct parameters to set? This is through something called a loss function, or an error function.[27] The loss function calculates the loss or total error between the outputs generated using the model, and the outputs in the training data. To begin the process, we provide training data to the model, and our training data gives an expected output, which is compared to the output of the model.[28] [29] After the first iteration of training is complete, the loss function calculates the total difference between the expected output and actual output, by calculating the sum of each individual difference.[30] [31] Two of the most common loss functions are mean squared error and cross entropy.[32] The former does just what it sounds like; it squares the error for each output value, and then calculates the mean. Cross entropy uses a similar approach but with a logarithmic function.

OPTIMIZATION

Now that we have a loss function, the next step is to choose an *optimization function*. The optimization function is used to determine the minimal loss; essentially, it *optimizes* for the minimum loss.[33] The optimization function will continually update the network so that the correct weights are chosen until the network's loss is at a minimum.[34] The optimization function has to look at the current loss, compare this to the new loss, and then update the weights accordingly. However, this gets really difficult when you have thousands or millions of neurons, and hence the need for special

[26] (Elgendy 2020, Ch. 2)

[27] Ibid.

[28] (Kelleher 2019, 97)

[29] (Artascanchez and Joshi 2020, Ch. 3)

[30] (Kumar 2019, Ch. 4)

[31] (Elgendy 2020, Ch. 2)

[32] Ibid.

[33] Ibid.

[34] Ibid.

optimization algorithms.[35] One really popular optimization algorithm that can help overcome this problem is called batch gradient descent. Basically, you look at the slope or rate of change of the line tangent to a given point of the curve, also known as the *gradient*, and then keep adjusting the model until you can minimize the gradient.[36] Then we can specify the amount to change each time, known as the step size, which allows us to fine-tune the algorithm to work best for any given situation.[37] Today, better variations of this algorithm exist, such as stochastic gradient descent and mini-batch gradient descent. The former uses only one instance of the data that changes randomly, rather than all of the data, while the latter divides the training into sections containing multiple instances, but not all of the data at once.[38] Both of these generally perform better than the standard batch gradient descent.[39]

BACKPROPAGATION

Once we have a loss function to determine the loss and we know the direction to adjust the weights given this loss, we still have to figure out how much to adjust each weight individually.[40] This is the function of the backpropagation algorithm. The backpropagation algorithm calculates the loss with respect to each specific weight, beginning at the last layer and working its way all the way up the network, until it reaches the initial input layer.[41] The algorithm accomplishes this functionality using partial derivatives. It distributes the loss among various weights by calculating the partial derivative relative to that weight, starting from the output layer and continuing until the start of the network.[42] However, one potential problem is that because the weights are calculated based on the partial derivative starting at the end of the network,

[35] (Elgendy 2020, Ch. 2)

[36] Ibid.

[37] Ibid.

[38] Ibid.

[39] Ibid.

[40] Ibid.

[41] Ibid.

[42] Ibid.

as the algorithm approaches the beginning, the value of the loss with respect to the weight diminishes until each weight is only given a very small effect. This is known as the problem of vanishing gradients.[43] Vanishing gradients makes choosing a good activation function much more complicated, because if the activation function grows to infinity as the value increases (or decreases), this will have a greater impact on the partial derivative and so will exacerbate the problem of vanishing gradients.[44] [45]

CONVOLUTIONAL NEURAL NETWORKS

Everything described thus far is at the heart of most modern artificial intelligence, specifically the field of deep learning, the most common artificial intelligence technology used today.[46] Basically, we have a sum of inputs that gather data, process that data through several hidden layers, applying both linear functions and nonlinear activation functions, and then finally output the data through a special output layer, which will be used to convert the result to the appropriate type.[47] Now in order for computers to sense different types of data, various add-ons and modifications to this model may be needed.[48] This paper describes the area of image recognition, also known as computer vision.[49] Computer vision, or the art of training a computer to observe patterns based on sight input,[50] makes use of a special type of neural network to enable image recognition and/or detection of visual objects, known as object detection.[51] This special type of neural network is called a convolutional neural network, or CNN.[52]

---

[43] (Elgendy 2020, Ch. 5)

[44] Ibid.

[45] (Elgendy 2020, Ch. 2)

[46] (Kelleher 2019, 2)

[47] Ibid, 76.

[48] Ibid, 159-60.

[49] (Elgendy 2020, Ch. 1)

[50] Ibid, Ch. 2.

[51] Ibid.

[52] Ibid, Ch. 3.

A convolutional neural network, or CNN, is a special type of neural net, which uses special sliding "layers,"[53] called convolutional layers, to slide over the object and thus be able to detect pixels at various locations of the image.[54] A convolutional layer consists of one or more *convolutional filters*[55], where each filter contains M x M neurons.[56] The filter, for example of size 5 x 5 neurons, slides over the image and the result of the image pixel values, convolved with the filter, is passed into what is called a feature map, basically a map specifying the presence of various features or visual elements, at a given location (see figure 2).[57]

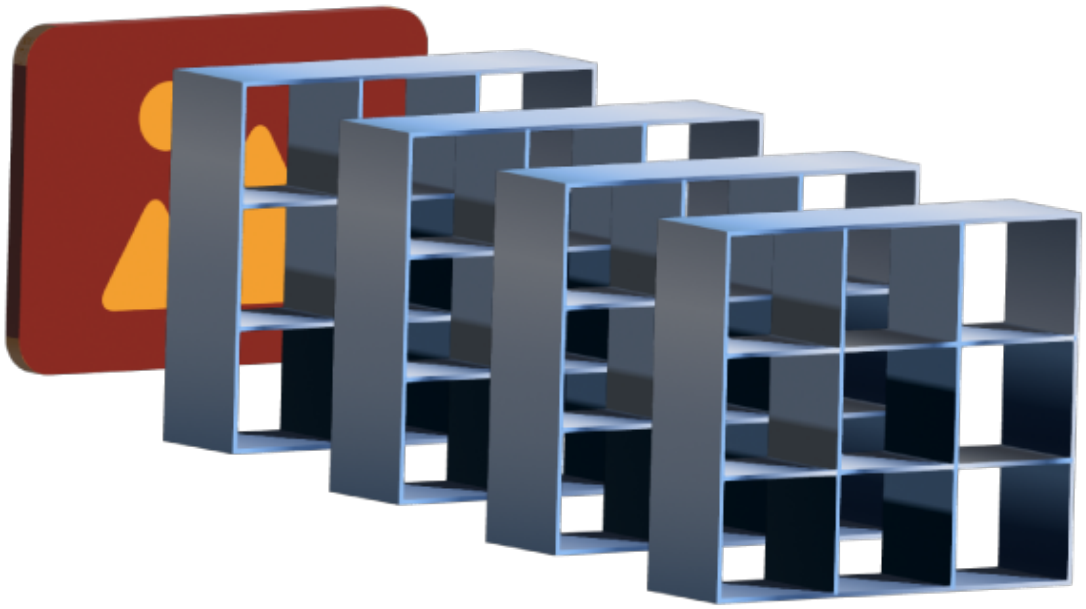


Figure 2. Four convolutional layers, with feature maps of size 3 x 3
Credit: Adobe Stock

Each feature map searches for the presence of a very specific element, like a line or a curve[58] and as the model goes deeper into the network, the features become less

[53] (Elgendy 2020, Ch. 3)

[54] Ibid.

[55] Ibid.

[56] Ibid.

[57] Ibid.

[58] Ibid.

general and more specific, until the system is able to detect the presence of specific objects.[59] Basically, layers are stacked against each other, until we have patterns within patterns within patterns, and the system is able to detect high-level, or human-friendly objects.[60] After all of these layers, there is one more layer. Known as a fully connected layer,[61] [62] this is just a normal layer and is used for the final step to classify the image; or for object detection, the last step before the network is brought into the next system.[63]

Now one problem with the process described above has to do with performance. Each convolutional filter stores pixel data for M x M values and as we increase the filter count, the amount of data that has to be stored in memory (VRAM) during training grows to be outrageous. Thus comes the concept of *pooling layers* (see figure 3).[64] A *pooling layer* applies a statistical function to the output from the previous convolutional layer to shrink the size (width and height) of the previous layer.[65] So as we progress, even though the feature map becomes very deep, its width decreases, and so we are able to decrease the amount of memory required to train the model.[66] Another concept is that of *strides*

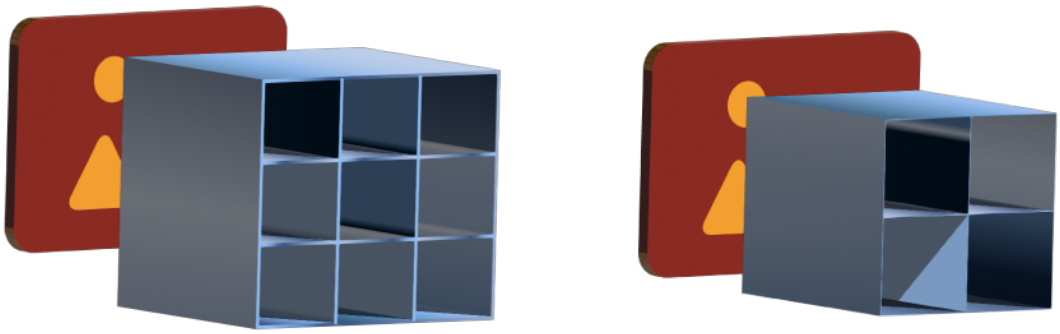


Figure 3. Effect of Pooling Layers
Credit: Adobe Stock

[59] (Elgendy 2020, Ch. 3)

[60] Ibid.

[61] Ibid.

[62] See section on neural networks.

[63] (Elgendy 2020, Ch. 7)

[64] Ibid.

[65] Ibid.

[66] Ibid.

and *padding*. The stride specifies the amount to slide the convolutional filter each time, while padding specifies whether we should add additional empty data or 0s around the border of the image.[67] Nowadays, there is a push to use strides and padding to decrease the image size instead of pooling layers.[68]

IMPROVING PERFORMANCE

The sections above describe the basic overview of a convolutional neural network. However, improving the performance of a convolutional network requires careful skill and observation, and is an ever-evolving field.[69] So what are some techniques for improving the performance of convolutional neural networks, in 2022? This section begins by discussing ways to measure performance, follows with a general introduction on designing accurate computer vision models and concludes by discussing some state-of-the-art techniques.

MEASURING PERFORMANCE

In order to know how to improve our model, we first have to be able to measure performance.[70] While loss is used to measure model accuracy during training, how do we measure the performance of the trained model itself? There are several performance metrics we can use. Accuracy measures how many predictions are correct, compared with total predictions.[71] However, in some cases, this can be deceptive.[72] Elgendy gives an example of a rare disease that is only found 0.1% of the time. A test that always returned negative would be 99.9% accurate, but nevertheless not testing anything.[73] Recall, or sensitivity tells us how many the model marked as positive which were positive,

[67] (Elgendy 2020, Ch. 3)

[68] Ibid.

[69] Ibid, Ch. 4.

[70] Ibid.

[71] Ibid.

[72] Ibid.

[73] Ibid.

over the amount marked as positive and incorrectly marked as negative.[74] Precision or specificity, on the other hand, is used to measure how many the model marked as positive that were positive, over the total number marked as positive.[75] Finally, the F-score shows us both precision and recall and is calculated by the following equation[76]: $\frac{2pr}{p+r}$. Used together, we can find the best benchmarks for our model, depending on the situation.[77]

GENERAL TIPS

All of the essential components of a deep learning model, like the loss, optimization, and backpropagation functions, as well as features specific to computer vision, such as convolutional layers, all have adjustable properties that we can tune to obtain optimal performance. These are known as hyperparameters,[78] and are set by the machine learning engineer, in contrast to parameters, which are set automatically by the system when the model is training.[79]

First and foremost, some of the hyperparameters that we can tune are the number of hidden layers and neurons in each layer.[80] While having more layers can produce better results, there are a couple of downsides. First, more layers or neurons can lead to longer training times, but also adding additional complexity will begin decreasing performance, a condition known as overfitting.[81] Generally, when our model starts underperforming when we add layers, we have two options. We can stop making the model more complex, or we can use additional techniques to prevent overfitting. One solution is to add dropout layers, which randomly turn off some of the neurons during

[74] (Elgendy 2020, Ch. 4)

[75] Ibid.

[76] Ibid.

[77] Ibid.

[78] (Kelleher 2020, 80)

[79] Ibid, 80, 82.

[80] (Elgendy 2020, Ch. 4)

[81] Ibid.

training.[82] Another tool we can use is called L2 Regularization. It adds a small penalty to the loss function that increases during each training iteration. This allows the model to update less frequently if there is a change, and so prevent overfitting.[83]

We can also choose which activation function to use. For example, the ReLU function will count negative values as zero, so a recent addition is to use a special type of ReLU function, called Leaky ReLU, so that negative numbers are accounted for.[84] We can then specify the rate at which the function "leaks." We can also specify hyperparameters for the loss and optimization functions. For example, we can specify which loss function to use, or which mini-batch size and learning rates to use with mini-batch gradient descent.[85] Finally, we can choose to specify how many times to train the data. These are known as iterations or epochs. The more epochs, the more thorough our model, but also the longer and the greater the risk of overfitting our data.[86] One especially sleezy option is to set a special callback function, that either automatically decreases our learning rate or completely stops training once our loss hasn't improved after the specified number of epochs.[87]

Finally, in addition to hyperparameters, there is one more thing, which has a huge impact on our data: preprocessing or preparing our training data. By conforming our data to some baseline, like the normal distribution, we can reduce the loss and help prevent overfitting.[88] One quick technique we should always use is to *normalize* our data.[89] Basically, we ensure that the data have small values all within the same range. This usually means that our data is converted to be in the range [0,1],[90] which is

[82] (Elgendy 2020, Ch. 3)

[83] Ibid.

[84] Ibid.

[85] Ibid.

[86] Ibid.

[87] Ibid.

[88] Ibid.

[89] Ibid.

[90] Ibid.

especially useful, since computers use the binary system. For convolutional neural networks, some preprocessing techniques we can use are augmenting different transformations to the images, converting images to grayscale, and resizing images to one size, prior to training.[91] These are also some examples of regularization techniques, which manipulate the data in some way to improve performance.[92]

FEW-SHOT OBJECT DETECTION

Modern deep learning models perform well, even in the area of computer vision, when given very large amounts of data.[93] However, there is the catch: massive amounts of manually labeled, parsed data is required in order to train the model. While this can be feasible for some models through crowdsourcing sites like Amazon Turk[94], it still remains a large hurdle before more widespread usage of CNN models is possible. One possible solution to this problem is Few-Shot Object Classification and Few-Shot Object Detection, collectively known as Few-Shot Learning.[95] Similar to how humans are able to quickly learn based on previous experience, Few-Shot Learning allows the system to learn how to identify objects after only a few examples, through utilizing previous knowledge.[96]

Few-Shot Object Detection operates on three essential components, the data, the model, and the algorithm. Prior knowledge is utilized for each of these categories to enable the system to learn based on only a few supervised examples.[97] [98] One technique is transfer learning, where a previously trained model is imported, but this can be

[91] (Elgendy 2020, Ch. 3)

[92] (Elgendy 2020, Ch. 4)

[93] (Antonelli et al. 2022)

[94] (Mueller and Massaron 2022, Ch. 11)

[95] (Antonelli et al. 2022)

[96] Ibid.

[97] Ibid.

[98] Whereas unsupervised learning is only based on finding patterns, supervised learning requires training data. See (Mueller and Massaron 2022).

difficult for object detection.[99] Another approach is distance metric learning, where samples are embedded to a low dimension space, and by comparing sample data and training data, the system is able to easily discover important features and distinguish between classes.[100] [101] Finally, meta learning teaches the model how to learn, using what is known as meta knowledge. Basically, the system is able to extract helpful information and then use this information to improve the model.[102]

ADDITIONAL REGULARIZATION TECHNIQUES

Regularization techniques were discussed earlier, in the section on general tips for improving performance. In particular, dropout layers and L2 regularization were mentioned, as well as data augmentation. However, in the past four years, numerous additional regularization technologies have been developed, a few of which are discussed here.[103]

DATA AUGMENTATION

The first category of data regularization techniques are new data augmentation methods. One technique is Cutout, a form of data augmentation where a portion of the image is removed, before being fed into the network for training.[104] Because this is supervised data, the augmented data is labeled, and so it has the effect of adding additional blemished samples to the training dataset.[105] A similar technique is RandomErasing, where the portions that are removed are replaced with random noise, instead of whitespace.[106]

[99] (Antonelli et al. 2022)

[100] Ibid.

[101] (Dor Kedem 2021)

[102] Ibid.

[103] (Gonçalves Dos Santos et al. 2022)

[104] Ibid.

[105] Ibid.

[106] Ibid.

Furthermore, a new technology called AutoAugment, automatically determines the appropriate augmentations for a dataset, and obtained excellent results on the CIFAR-10, CIFAR-100 and ImageNet datasets, and could even be transferred to additional datasets.[107] Moreover, the biggest issue, the amount of time required to train, is largely fixed with a newer version called Fast AutoAugment.[108] Yet another technology, RandAugment looks at the 14 most common policies from AutoAugment and applies those, achieving remarkable results.[109] A few other augmentation regularization techniques are MixUp, CutMix, CutBlur, BatchAugment, and FixRes. MixUp and CutMix combine two images and help the model correctly decipher the probability of it being one of the two images.[110] CutBlur helps train super-resolution images as well as reconstruct data for blurred images.[111] BatchAugment splits the GPU memory when training, so half of the memory can be used for augmented images.[112] Finally, FixRes uses a larger resolution for testing than for training, which can improve performance by making the model more reliable and use less time.[113] Finally, performance gains can also be reached by combining augmentation methods.[114]

INTERNAL STRUCTURE CHANGES

Another category of regularization techniques works by changing the internal structure of the neurons during training. For example, Dropout randomly shuts off some neurons when training each epoch.[115] A new version of Dropout, known as MaxDropout, shuts off neurons based on their activation above a given level. The more a neuron

[107] (Gonçalves Dos Santos et al. 2022)

[108] Ibid.

[109] Ibid.

[110] Ibid.

[111] Ibid.

[112] Ibid.

[113] Ibid.

[114] Ibid.

[115] Ibid.

exceeds a given threshold, the greater the probability of the neuron being turned off.[116] DropBlock removes correlated regions created by the CNN system, and TargetDrop can be combined with DropBlock to remove only appropriate regions.[117] Moreover, some additional strategies are AutoDrop, which automatically determines the best drop design, Shake-Shake, which assigns different weight values for each forward and backward (backpropagation) pass, and ShakeDrop, a smarter, more capable version of Shake-Shake.[118] Finally, a special technique, called Manifold MixUp, performs mix-up after training has begun, i.e., mixing up images after the input layer.[119]

LABEL REGULARIZATION

Finally, some of the techniques mentioned above are also label regularizers. Namely, they control how the label data is presented, to help the model train more effectively.[120] For example, MixUp averages the values of the labels.[121] This is also known as label smoothing.[122] Other label smoothing techniques are also commonly used, like one-hot encoding, that reformats labels to help prevent overfitting.[123] [124] There are a few other techniques as well, which aim to improve label smoothing. Some of them are Two-Stage Label Smoothing, which only uses label smoothing for part of training, Structural Label smoothing, which uses Bayes' Theorem to calculate error, which is then used to determine smoothing values, and JoCor, which trains two networks at the same time and then compares the two.[125] Unfortunately, no regularizer is best for all circumstances. The

[116] (Gonçalves Dos Santos et al. 2022)

[117] Ibid.

[118] Ibid.

[119] Ibid.

[120] Ibid.

[121] Ibid.

[122] Ibid.

[123] Ibid.

[124] (Elgendy 2020, Ch. 3)

[125] (Gonçalves Dos Santos et al. 2022)

best approach is to attempt combining various regularizers, and adjusting them for the problem at hand, until the best results are obtained.[126]

MODERN CNN BLOCKS

Lastly, in recent years there has been a huge development in CNN models, replacing standard convolution layers with special convolution blocks designed to improve performance.[127] One example is with the ResNet architecture, which uses special residual blocks to prevent the vanishing gradients problem that was discussed earlier.[128] Basically, there are "shortcut connections," which enable "residual" connections, i.e. shortcuts to earlier layers in the network which are connected directly to the output.[129] More recently, EfficientNet uses multiple types of these Bottleneck Blocks or BBs to achieve both high accuracy and low computation cost.[130] [131] This also enables the model to scale well even for very large networks.[132] One type of block used by EfficientNet is the point-wise convolution layer, which uses 1 x 1 filters, reducing the total number of operations.[133] Depth-wise convolution blocks are also employed, which divide the input (pixel) channel(s), into additional channels.[134] [135] This helps improve performance since parallelization or parallel processing can be employed.

However, one setback we are facing is the issue of finding a CNN accelerator that improves performance for some of these recent, more efficient blocks.[136] For example,

---

[126] (Gonçalves Dos Santos et al. 2022)

[127] (Lee et al. 2022)

[128] (He et al. 2015)

[129] Ibid.

[130] (Lee et al. 2022)

[131] (Tan et al. 2019)

[132] Ibid.

[133] (Lee et al. 2022)

[134] Ibid.

[135] One input channel is used for grayscale images, while color images have three channels. See (Elgendy et al. 2020, Ch. 1).

[136] (Lee et al. 2022)

the depth-wise convolution blocks do not reuse the input feature maps between output channels.[137] However, this can be a problem, because it prevents the blocks from being sped up via conventional CNN accelerators.[138] One recent proposal is the MVP CNN Accelerator, which features matrix, vector, and processing units close to memory. Utilizing a concept similar to that used in Apple's M1 Chip[139], this enables the accelerator to optimize the performance of both computationally intensive layers and memory-intensive layers, and thus be effective for all types of modern convolutional networks, at only a small performance penalty.[140]

CONCLUSION

Modern artificial intelligence has gone a long way to become the backbone of computer-driven cutting-edge technologies. A simulation of human intelligence and based on the human brain, artificial intelligence seeks to imitate human behavior, and accomplishes this primarily through the use of what is known as a neural network or neural net. Containing millions of neurons, the system automatically finds the optimal values for these neurons through the use of loss, optimization and backpropagation functions. In the area of computer vision, artificial intelligence is applied primarily through the use of convolutional neural networks, which use a special convolution layer featuring a convolution filter that passes over the input and extracts data into feature maps. These feature maps start very generally and slowly grow more specific until the computer is able to recognize high-level objects.

Some ways of improving convolutional neural networks include the use of dropout layers, L2 Regularization, and normalization of data. Adjusting the hyperparameters allows us to fine-tune the model to perform best to our situation. Moreover, Few-Shot Learning allows the computer to easily learn new images and adapt to new scenarios

[137] (Lee et al. 2022)

[138] Ibid.

[139] (Apple 2020)

[140] Ibid.

through the use of previous knowledge, utilizing technologies like transfer learning and meta learning. Furthermore, numerous data regularization techniques exist, including several new data augmentation techniques, various methods that change the structure of the data, and lastly, various ways to obtain optimal performance through the use of label smoothing.

Finally, modern convolutional networks use special blocks instead of standard convolution layers, which help increase performance, including precision and training time. One state-of-the-art system utilizes a special accelerator that would enable these new blocks to be further optimized, just as conventional accelerators are able to optimize standard convolution layers. All of these techniques improve the performance of CNN networks, and are likely to be expanded upon in the near future, to pave way to new innovations and technologies. In the meantime, numerous artificial intelligence technologies and ever-growing state-of-the-art innovation make the current era one of the most exciting times to be a computer programmer, and one of the best for a machine learning engineer.

REFERENCES

Mueller, John Paul, and Luca Massaron. 2022. *Artificial Intelligence for Dummies. 2nd edition.* Hoboken, NJ: John Wiley & Sons, Inc.

Artascanchez, Alberto, and Prateek Joshi. 2020. *Artificial Intelligence with Python. 2nd edition.* Birmingham, UK: Packt Publishing.

John D. Kelleher. 2019. *Deep Learning*. Cambridge: The MIT Press. https://doi.org/10.7551/mitpress/11171.001.0001

Mohamed Elgendy. 2020. *Deep Learning for Vision Systems.* Shelter Island, NY: Manning Publications.

Rahul Kumar. 2019. *Machine Learning Quick Reference*. Birmingham, UK: Packt Publishing.

Antonelli, Simone, Danilo Avola, Luigi Cinque, Donato Crisostomi, Gian Luca Foresti, Fabio Galasso, Marco Raoul Marini, Alessio Mecca, and Danielle Pannone. 2022. "Few-Shot Object Detection: A Survey." *ACM Computing Surveys*. https://dl.acm.org/doi/pdf/10.1145/3519022

Dor Kedem. 2021. "PyData Global 2021 Introduction to Distance Metric Learning." Accessed April 25, 2022. https://github.com/kedemdor/metric-learning-talk/blob/main/PyData_Global_2021_Introduction_to_Distance_Metric_Learning.ipynb

Gonçalves Dos Santos, Claudio Filipi, and João Paulo Papa. 2022. "Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks." *ACM Computing Surveys.* https://dl.acm.org/doi/pdf/10.1145/3510413

Lee, Sunjung, Jaewan Choi, Wonkyung Jung, Byeongho Kim, Jaehyn Park, Hweesoo Kim, and Jung Ho Ahn. 2022. "MVP: An Efficient CNN Accelerator with Matrix, Vector, and Processing-Near-Memory Units." ACM Computing Surveys. https://dl.acm.org/doi/pdf/10.1145/3497745

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, Jian Sun. 2015. “Deep Residual Learning for Image Recognition.” https://arxiv.org/pdf/1512.03385.pdf

Tan, Mingxing, and Quoc V. Le. 2019. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” *International Conference on Machine Learning.* https://arxiv.org/pdf/1905.11946.pdf

Apple Press Release. 2020. “Apple Unleashes M1.” Accessed April 25, 2022. https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/