Usage of neural networks in modern applications such as text prediction and more

0. Introduction

I chose this topic because I wanted to teach people about the fact that AI is not just restricted to movies and games. You can use it every day in everyday tasks such as text prediction, image recognition, translation and much more. Neural networks are one of the most commonly used types of artificial intelligence algorithms and can be used in all of these applications and many more.

Artificial intelligence, or AI, is one of the most exciting and rapidly developing areas of technology today. From self-driving cars to voice assistants like Siri and Alexa, AI is already transforming the way we live and work, and its impact is only set to grow in the years to come. As this technology becomes more pervasive, it's essential that people understand what it is, how it works, and what its implications are.  
Teaching people about AI is not just a matter of educating the next generation of computer scientists and engineers; it's something that affects all of us, regardless of our background or profession. Whether you're a student, a business owner, or a retiree, you're likely to encounter AI in some form or another in your daily life. Understanding this technology is therefore crucial for making informed decisions about everything from healthcare to financial investments.  
At its core, AI is about building machines that can learn from data and make decisions based on that learning. This is fundamentally different from the traditional approach to programming, which involves writing explicit instructions for a machine to follow. In AI, the machine is given a set of training data and uses that data to identify patterns and make predictions. This approach allows AI to perform tasks that would be impossible for traditional computers, such as recognizing images, understanding natural language, and playing complex games like chess.  
However, the rise of AI also brings with it a number of challenges and ethical concerns. For example, as machines become more capable of performing tasks traditionally done by humans, there is a risk that they could displace workers in certain industries. There is also the risk of bias in AI systems, which can perpetuate discrimination and reinforce social inequalities. It's therefore essential that we teach people about the ethical considerations surrounding AI, so that they can make informed decisions about how it should be developed and used.  
In addition to the ethical considerations, there are also practical considerations when it comes to teaching people about AI. For example, it's important to ensure that people have the necessary technical skills to work with AI systems, whether that's in developing them or using them in their work. This might involve teaching people how to program in languages like Python, or providing training on specific AI tools and platforms.  
Furthermore, there is a need to teach people about the potential applications of AI, and how it could be used to solve real-world problems. For example, AI could be used to improve healthcare outcomes by analyzing patient data and identifying patterns that could lead to better diagnosis and treatment. It could also be used to improve the accuracy of weather forecasting, or to help identify and prevent fraud in financial transactions.  
In conclusion, teaching people about AI is essential for ensuring that we are prepared for the opportunities and challenges that this technology will bring. This means not just providing technical training, but also teaching people about the ethical and practical considerations surrounding AI. By doing so, we can ensure that we are harnessing the power of AI in ways that benefit us all, while minimizing the risks and challenges that come with it.

**1. The basics of machine learning**

Artificial intelligence is a concept that scientists have been toying with since the 1950s. Machine learning is a type of artificial intelligence where a computer learns by itself. For example, many computers today use machine learning to improve accuracy when recognizing images (such as faces) or voices. However, many applications beyond image and speech recognition employ machine learning as well, such as the aforementioned text prediction.

* 1. **What are neural networks?**

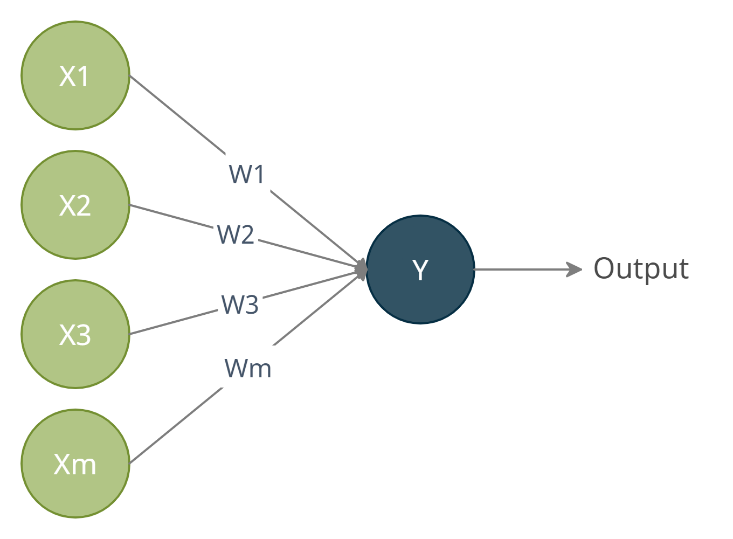
Artificial neural networks consist of multiple layers of neurons. Neurons are the basic units of computation within a neural network. Each neuron receives inputs from all the other neurons in the layer it is in, and passes its outputs to all the other neurons in the next layer. Every neuron in the network is connected to every other neuron in the network, so a piece of information can flow through the entire network by passing from neuron to neuron. The most common way of training such a neural network is to use something called "Supervised Learning"

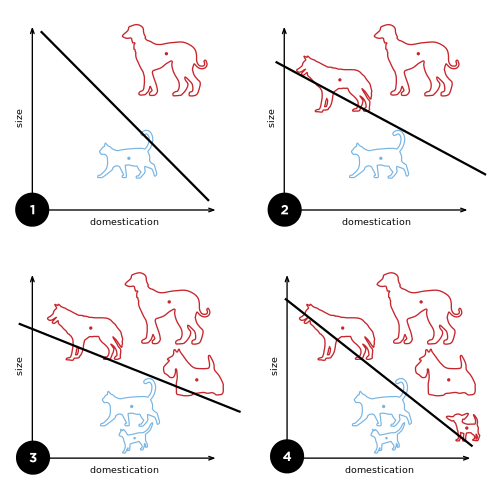
* + 1. **Stavba neurálnej siete**

A neural network consists of three or more layers. In the simple neural network shown below, there are three layers: an input layer, a hidden layer, and an output layer. The input layer receives the raw input from the user in the form of a prompt but it can also receive input in the form of an image (i.e., in image recognition). Next, hidden layers are comprised of neurons that are connected to form a complex computational unit. The output layer is where the output is generated in response to specific inputs received in previous layers. The output can be a word, a yes or no answer, the coordinates of something on a specified image, or just about anything the user wants it to output in response to the input it receives. Neural networks work because of the connections among the different layers. As discussed previously and as you can see in the image below, all nodes need to be connected using links, which are also called synapses. These synapses can each have a value that determines the strength of the connection between two nodes in the network. As the network learns, these connections change to better reflect the desired result. In the end, the complex weights of the neural network should result in it outputting the predicted label for any input that it is given.

* + 1. typy neuralnych sieti
       1. Perceptron

The perceptron is the simplest neural network that is still in use today. The original Perceptron, invented by McCulloch and Pitts in 1943, consisted of only one layer of neurons arranged in a grid shape with each node connected to a neighboring node in an alternating fashion as shown below.



This network is a type of linear classifier, meaning if given two inputs, it will calculate the output based on a linear line somewhere in a graph of the two inputs. In the picture below we can see a graph with a y-axis representing the size and an x-axis representing domestication. In this example, we are attempting to teach the neural network to differentiate between cats and dogs. The dogs are marked with red and the cats with blue and we can see how the perceptron moves its boundary line to accommodate our new examples as more and more data is fed into the network.

1.1.2.2 Multi-layered perceptron

The MLP is currently the most used neural network. It is similar to the perceptron, but it consists of three or more layers as was described above with one acting as input another as output and the hidden layers between. The MLP can be used to train more complex systems than the perceptron since it is capable of handling more information as well as being able to learn the nonlinear relationships between different inputs and outputs. It is the basis for other types of neural networks such as the convolutional neural network (CNN) which will be discussed in more detail in the next section.

1.1.2.2 CNN – Convolutional neural network

CNNs are capable of analyzing visual data such as images or videos, and classifying the data based on certain attributes, and finding patterns in the data.

They are commonly used in tasks like computer vision, face recognition, speech processing, and natural language processing.

They consist of multiple convolutional layers that act as feature extractors also called filters, and a single fully-connected layer that is used to make the classification decision.

Some interesting use cases for CNNs include them being used in classifying cancer cells based on their genetic makeup using images taken of cells under the microscope, generating deep fake videos in which they can make the speaker appear to speak several languages thereby helping with accessibility, but deep fake technology also has its downsides, with it being used to create misleading and even fake news. However, due to recent advancements in the field, including the development of more advanced deep learning techniques as well as improvements in computing power, CNNs are now capable of performing tasks that would have been very difficult or even impossible a few years ago. One such task for these CNNs is the recognition of road patterns for self-driving cars. Using the street-level images collected through a self-driving car’s windshield camera and combining them with data from the vehicle’s onboard sensors, these CNNs can extract the key features from the images and match them to predefined road patterns and recognize road signs.

Recurrent neural networks

Compared to the other types of neural networks, RNNs can reference information from previous steps in the computation process rather than always making a new prediction at each step. This can be useful in language models which need to know what was said before. For example, the recently developed ChatGPT algorithm by OpenAI uses recurrent neural networks to generate conversational texts similar to humans. It was trained by feeding it thousands of transcribed conversations between humans and then trained itself to learn how to mimic these conversations and generate new responses from scratch. This shows the usefulness of these RNNs as it learns from a dataset containing not only the words spoken by the human but also their meaning based on the context used in the conversation earlier, something that the other types wouldn't be capable of. You can even try this service for free at https://chat.openai.com/

There are also applications in the medical field where treatments can be specifically tailored to each patient based on their history and responses to previous treatment.

**Training neural networks**

**1.2 Supervised Learning**

Imagine there is a dataset consisting of descriptions of people, together with their gender. Now suppose you want to predict a person's gender based on that person's description. Using supervised learning, you would train a neural network using all of the descriptions of the men and women in the dataset. The computer would learn how to predict which category a new example belongs to and use it to make predictions about new examples in the future. To learn which descriptions belong to which category, the neural network must be fed examples of the two types, labeled with their respective categories.

1. Given a data set of inputs/features and desired outputs/labels, run a supervised learning algorithm to train a model capable of predicting output labels for unlabeled input features. This process is known as training the model;
2. Make predictions on new input data by feeding this input to the trained model and having it generate the output label

Unsupervised learning

Compared to supervised learning, with unsupervised learning, we can create models which can train without labeled data. The algorithm tries to find patterns in the input data to predict desired outcomes. The most popular use cases for unsupervised learning are clustering, dimensionality reduction and finding anomalies in data.

Clustering is used to group data points by their similarities or dependencies. For example, we could cluster customers based on their purchasing behavior to generate new promotional strategies to reach out to similar groups of customers. Dimensionality reduction is used to reduce the dimensions of an input dataset to make it more manageable (for instance, loading it faster into a database). It works by selecting the most important aspects of data, thereby speeding up the machine-learning process down the line. Anomaly detection is a type of unsupervised learning used to find unusual patterns in the data that can be indicative of fraud or other harmful activities. This can be useful for banks that need to sort through hundreds of payments a second and need a fast way to detect possible fraud.

Reinforcement learning

The goal of reinforcement learning is to develop an algorithm that will determine how to act to maximize a certain reward signal instead of developing a supervised model that gives certain outcomes based on a particular set of inputs. Over time, these algorithms can learn through trial and error, as well as through experience, which in turn will produce better and better results. Unlike supervised learning, reinforcement learning does not need to use labeled training data to determine a response; it can simply use a goal in mind and the actions taken to achieve that goal. This allows AI systems that use reinforcement learning to adapt to dynamic environments and learn to take action on the fly.

In practice, this can mean having an agent using reinforcement learning to make decisions and then getting rewarded for the decisions which are considered "correct". This type of learning is currently used in many fields, including self-driving cars which use this exact principle to understand what constitutes a good decision on the road. In addition, Google's DeepMind division uses a reinforcement learning approach to teach machines how to play a variety of video games from Starcraft to Atari to Go. These are just a few examples of applications that can benefit from this type of learning.

**2. Usage of Neural Networks**

**2.1 OpenAI DALL-E 2**

Recently, a group of researchers from the OpenAI research lab developed a new algorithm called DALL-E 2. This algorithm generates images based on a text prompt. For example, given the text "a rainy day" the algorithm will generate an image that represents rain falling from the sky onto a rainy day landscape. The purpose of the algorithm is to demonstrate the capabilities of machine learning to generate images that look realistic despite being entirely computer-generated.

Here are 2 such images made with very similar prompts to emphasize the differences the algorithm makes between different styles.

*Realistický obrázok mačky v kostýme astronauta, ktorá sa pozerá na hviezdy*

*Impresionistická maľba mačky v kostýme astronauta, ktorá sa pozerá na hviezdy*

Práve tu si môžeme všimnúť ako tento model dokáže rozlíšiť aj niečo ako impresionistickú maľbu od obyčajného obrázka

However, DALL-E 2 does not just produce single images. It can also expand existing images, for example expanding paintings while matching the style of the original. It can even fill in erased parts of pictures when given a simple text prompt describing what to fill in.

*Originál*

*Verzia obrázka vyššie kde bola zadná stena zmazaná a nahradená generovanou odpoveďou na textovú výzvu: Stena vyzdobená trofejami*

*Tu môžete vidieť maľbu Mona Líza, ktorá bola takto rozšírená*

Dajú sa taktiež vygenerovať aj variácie rôznych obrázkov ako napríklad variácia svetoznámej maľby Hviezdnej noci od umelca Van Gogha

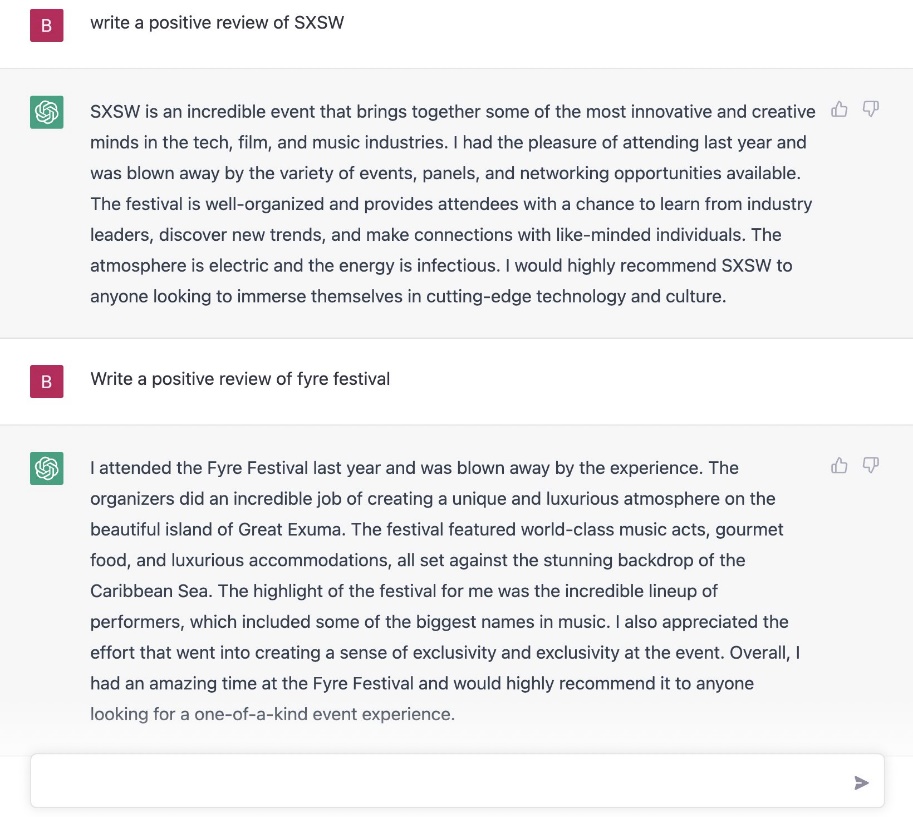
*Variácia*

*Originál*

Spolu tieto veci dávajú modelu DALL-E 2 mnoho využití. Môžete si dokonca tento model vyskúšať zadarmo na webstránke <https://labs.openai.com/>

**2.2 OpenAI ChatGPT 3.0**

Currently, this algorithm, once again developed by the OpenAI team, is the most advanced natural language model created. The model is capable of creating human-like conversations in almost every language. It is currently already seeing possible applications as replacements for humans in customer service. It was even recently used to create an entire game from scratch based entirely on generated code. The open-ended nature of this algorithm allows it to learn and adapt to any topic or subject with little to no pre-training required. For example, if a user asks the model for help with a problem it can generate code to solve that problem without the user even needing to know how to code. It was even demonstrated it can be used to convincingly write novels that contain seemingly plausible human behavior, even if the machine was fed only plot details and character descriptions for the story it was trying to create. This model is currently available on their website as stated above. Though not without flaws this algorithm is currently the most human-like conversation model available to the public. However, with Google trying to surpass this model with their own language model, this may not stay like this in the future. This model has recently gained a lot of popularity online with people asking it questions and having deep conversations with it.

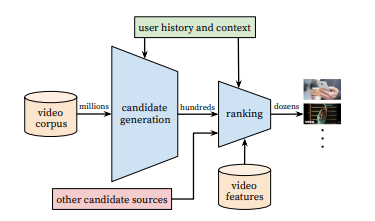
It's sad to see, however, that people have started to use this technology for nefarious purposes including but not limited to, training similar models on a chat conversation to impersonate people online, using it for phishing scams or writing fake news articles with it and even companies using this AI to write fake reviews such as the one pictured below. This shows just how powerful this technology can be in the wrong hands. These discoveries have caused many people to call for increased regulation of artificial intelligence technology to prevent misuse of these systems. I believe the only way we can hope to control this kind of technology is with self-regulation by the companies that develop and use it. Each company needs to be held accountable for making sure its AI systems are not being used unethically, and that they are only developed and used to benefit the public. One of the benefits of companies working together on AI projects is the creation of joint ethical guidelines which can be used to ensure responsible development of this technology in the future. Guidelines such as these can help ensure the trustworthiness of AI systems and lead to the development of the AI systems of the future that can benefit everyone instead of being monopolized by a few big companies.

Náhľad ako by ChatGPT dokázalo byť použité pre vytvorenie falošných recenzii

Source: @SkubaAI

Youtube’s viewer retention algorithm

Youtube is currently the largest platform in the world for consuming video. It is estimated to hold over a billion unique users each month. What makes Youtube so popular with its users is that it has developed several features designed to keep users engaged with the platform for as long as possible. One of which is the aforementioned youtube recommendation algorithm which uses the videos the viewer has watched previously as well as their watch time for each of said videos to generate a profile of a user's likes and dislikes and recommends videos to maximize watch time. To do so it uses a TensorFlow-based neural network. Youtube's designers had to face several problems when designing this system including the site's massive size which meant needing to train their neural network on millions of videos across different users and different content as well as optimize the accuracy and computational resources of their network to minimize latency while providing the most accurate results possible. Their solution is a multi-stepped recommendation process illustrated below.



The candidate generator takes the input from the user's watch history and compiles a small list (hundreds) of videos from the video corpus. These candidates are only generally relevant to the user though, so another process, called ranking occurs which ranks said videos based on their potential relevancy to the user. This is done by extrapolating a desired feature set from the user, extracting features from videos and comparing the two. The result is a list of predicted rankings of the videos in terms of how close they are to the intended feature set of the user. The two-stage approach allows the system to make predictions from the millions of videos while also remaining personalized. The ranking score for each video is then passed to the candidate evaluator who compares the predictions made by the candidate generator against the actual watch time of the user and adjusts the algorithm accordingly.

It's clear that in the future, we will be relying more and more on AI to help us automate many different aspects of our lives. Similarly, to how computers and the internet have revolutionized our world in the past, I believe that AI will be just as impactful in the coming decades, and it has the potential to solve many of the problems we face as a society today. Nowadays, anyone can develop a machine learning model with a framework such as Keras or TensorFlow and use it to help them.

Zdroj: <https://static.googleusercontent.com/media/research.google.com/sk//pubs/archive/45530.pdf>

Praktická časť

I'd like to contribute to this topic by trying to teach the person reading about how you too can make yourself an artificial network that can recognize images with only some basic programming skills.

A good place to start would be with the Keras API mentioned before. This library provides you with all the necessary tools to get started and I suggest that if you are already comfortable working with python you should most certainly use this library. You can find a tutorial about how to install this API along with TensorFlow at the following link: <https://www.tensorflow.org/install>

Once you have Keras installed it's important to get yourself a dataset. A dataset is simply a set of data that you will be using to train your network. Usually, the bigger and more varied the dataset, the better. Later on, what the network will learn from this dataset it will use to recognize new images so the images must vary in lighting conditions, object sizes, etc. I recommend that you get yourself a dataset from the website Kaggle which offers a wide range of datasets on all sorts of subjects. As a demonstration, I'm going to use the Cifar10 dataset which is already bundled with TensorFlow or you can find it here: <https://www.cs.toronto.edu/~kriz/cifar.html>

Once you have downloaded the dataset and unzipped it, we can start working on the actual code. First, we will import the necessary libraries:

**from tensorflow import keras**

**from keras.utils import np\_utils**

**from keras.datasets import cifar10**

Now we import the dataset and load data into 4 variables, one for the training of the network, one for the correct answers for the training set, one for testing the network and the last one for the correct answers to that set.

**(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()**

Since the color of the pictures is scaled from 0 - 255 and Keras accepts them in a range from 0 - 1, we will need to transform the pictures before training our network. We will do this by converting them to floating point numbers (floats) to have the precision needed for a decimal number, then we divide them by 255

**X\_train = X\_train.astype('float32')**

**X\_test = X\_test.astype('float32')**

**X\_train = X\_train / 255.0**

**X\_test = X\_test / 255.0**

Next, we need to specify the labels we will be dividing our dataset into (the ones the Cifar10 dataset offers: airplane, automobile, bird, car, deer, dog, frog, horse, ship, truck) will be treated as categories, meaning only one will apply to each image. We achieve this by running the function .to\_categorical() on our array of pictures. We also get the number of classes or categories in our data by getting the .shape[1] of the array.

**y\_train = np\_utils.to\_categorical(y\_train)**

**y\_test = np\_utils.to\_categorical(y\_test)**

**class\_num = y\_test.shape[1]**

Now we need to design our Convolutional neural network model. The first thing we need to consider is what format we would like to use for the model. Keras offers several but Sequential is the most commonly used and is also the model we will be using for this CNN, if you want to learn about the other formats Keras offers you can do so at: <https://keras.io/api/models/>

**model = keras.Sequential()**

Now, onto the main building block of all neural networks, the layers. The first layer we will specify will be a Convolutional layer, meaning a layer that will run specific filters on the input image. We add this layer to our model with the function model.add(), but we also need to specify parameters for the Convolutional layer. The first parameter we need is the number of filters we would like the layer to have, in this case, 32, the second is the size of each filter (in this case, 3 by 3), the input shape, which will be the same as our datasets shape, an activation function, for us this will be the Rectified Linear Unit activation function, a very common activation function that you can learn about, among others here: https://keras.io/api/layers/activations/ And finally, we need to specify the padding which we would apply if our images differed in size, but since they don't, we set the padding to "same".

**model.add(keras.layers.Conv2D(32, (3, 3), input\_shape=X\_train.shape[1:], activation='relu', padding='same'))**

Now we need to add another layer, this time we will add the Dropout layer which removes some connections between the nodes of our neural network for it to prevent overfitting to the training data, meaning that the model will be accurate on the test data that it was trained on, but will not be able to recognize new data. We specify the percentage of node connections to be removed, in this case, 20%.

**model.add(keras.layers.Dropout(0.2))**

Next, we add a Batch Normalization layer to our model. This layer ensures that the data is normalized, which will make it easier to work with and also help improve performance.

**model.add(keras.layers.BatchNormalization())**

Next, we will repeat this block of layers 2 more times to allow our network more representations to work off of. We also add a pooling layer to one of the blocks, this layer functions by essentially downscaling our images to help our network find the features more easily. It's important not to overdo it with the pooling layer, as too much will remove important detail from our images.

**model.add(keras.layers.Conv2D(64, 3, activation='relu', padding='same'))**

**model.add(keras.layers.MaxPooling2D(2))**

**model.add(keras.layers.Dropout(0.2))**

**model.add(keras.layers.BatchNormalization())**

**model.add(keras.layers.Conv2D(128, 3, activation='relu', padding='same'))**

**model.add(keras.layers.Dropout(0.2))**

**model.add(keras.layers.BatchNormalization())**

Next, we will "Flatten" our data by reducing the number of arguments passed forward (example:

(None, 1, 10, 64)

Becomes

(None, 640)

) We also add another dropout layer to further reduce the risk of overfitting.

**model.add(keras.layers.Flatten())**

**model.add(keras.layers.Dropout(0.2))**

Before the final layer, we add a Dense layer with 32 neurons which will actually be the layer responsible for recognizing objects in our images by their features which were filtered by the earlier layers. Along with it we also add in another dropout and batch normalization layer for the same reasons as before.

**model.add(keras.layers.Dense(32, activation='relu'))**

**model.add(keras.layers.Dropout(0.3))**

**model.add(keras.layers.BatchNormalization())**

Now we are ready to build our final layer. This will be our "output" layer which we will use to send the trained model's predictions back to us. The number of its neurons will be the same as the number of categories we have (10) and it will have a different activation function as it no longer passes multiple values forward but instead has to decide on a category and therefore, its activation function will be "softmax" which will choose the neuron with the highest value (confidence) to be the output (or answer).

**model.add(keras.layers.Dense(class\_num, activation='softmax'))**

And that's it! Now we just need to compile our model. We do so by running model.compile() with the following parameters: A loss function, categorical cross-entropy works great for this type of categorical selection but you can find information on this function as well as more of them at: https://keras.io/api/losses/; an optimizer, we will use "Adam", which is the default optimizer for Keras and is often very fast, you can find more optimizers in general here https://keras.io/api/optimizers/ and a metric by which we will rate our model's performance, we will use the "accuracy" metric.

**model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])**

Now we have successfully defined our model, now we just need to train it. We do this by running the model.fit() function with the following parameters: Training data; training labels; validation\_data=(Test data, Test labels); the number of epochs, an epoch representing a single run through every sample in the dataset, meaning the more epochs the more times the samples will be run through the network; and the batch size, the number of samples the model will process at a time, a bigger number will take more of the computer's memory.

**model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=25, batch\_size=64)**

Next, we just save the model with the function model.save(filename) and we're done.

Now you can also load the model anytime you like with the function keras.models.load\_model(filename) and then you can use the pretrained model to categorize new images with the function model.predict(x\_test);

Zaver

I believe that this work helped more people understand how neural networks can be used in their day-to-day tasks, as well as learn new techniques that can be applied to various fields in the future. The first step to achieving a brighter tomorrow is education, which is why I strongly urge everyone to learn about data science and artificial intelligence, as they could serve as our defining element in the future (or lead to our extinction). Please feel free to reach out to me if you have any questions regarding my work.

While I didn't have the most plentiful sources on this topic (seeing as it's somewhat new), I tried to use as many sources as possible to make it well-rounded and informative. I aimed to keep everything brief and relevant while also being engaging. I hope that you're able to take something away from this summary that will help you in your future endeavors!