CHAPTER 1: DEEP LEARNING

**Table of contents**

1 introduction………………………………..

2 definition of deep learning…………….

3 deep learning applications……………..

4 deep neural network architectures…..

4.1 Convolutional Neural Networks

4.1.1 Introduction to CNN

4.1.2 How CNNs Work

4.1.3 Common Architectures of CNNs

4.1.4 Applications of CNNs

4.2 Recurrent Neural Networks

4.2.1 Introduction to RNN

4.2.2 How RNNs Work

4.2.3 Common Architectures of RNNs

4.2.4 Applications of RNNs

4.3 Transformers

4.3.1 Introduction to Transformers in Deep Learning

4.3.2 The Transformer Architecture: key components and functionality

4.3.2.1 Encoder and Decoder Stacks

4.3.2.2 Attention

4.3.2.2.1 Scaled Dot-Product Attention

4.3.2.2.2 Multi-Head Attention

4.3.2.2.3 Applications of attention in transformer model

4.3.2.3 Position-wise Feed-Forward Networks

4.3.2.4 Embeddings and Softmax

4.3.2.5 Positional Encoding

4.3.3 Advantages of Transformers

5 Conclusion

1. **Introduction :**

Artificial Intelligence (AI) refers to the development of intelligent systems that can perform tasks that normally require human intervention. Machine Learning (ML) is a subset of AI that allows systems to automatically improve their performance through experience. Deep Learning (DL) is a subset of both AI and ML that uses artificial neural networks to learn and make decisions like humans. Unlike ML, DL can process and analyze large volumes of unstructured data, such as images, speech, and text, to generate more accurate results. In this chapter, we will explore the fundamental concepts of DL, its architecture, and its applications in various fields.

1. **Definition of deep learning :**

Deep learning (DL) is a subset of machine learning (ML) that uses multilayer artificial neural networks to model and solve complex problems. These networks are designed to simulate the behavior of neurons in the human brain, enabling them to process and learn from large amounts of unstructured data.

Deep learning algorithms automatically learn to recognize patterns and features in data by analyzing and adjusting the weights and biases of network interconnected nodes. This process, called training, involves optimizing network parameters to minimize errors and improve accuracy. [1]

Deep learning has grown in popularity in recent years due to its ability to process large and diverse datasets, achieve cutting-edge performance in many fields, and achieve breakthroughs in areas such as computer vision, natural language processing, and speech recognition. [2]

1. **Deep learning applications:**

Deep learning (DL) is a powerful tool that has revolutionized many fields by delivering unprecedented accuracy and efficiency in processing complex data. One of the main strengths of DL is its ability to automatically extract relevant features and patterns from large and diverse datasets without explicit feature engineering. This makes deep learning particularly suitable for applications such as image and video processing, natural language understanding, speech recognition, and autonomous decision-making. [1]

For example, DL has made significant progress in image and video recognition tasks such as object recognition, face recognition, and scene understanding. Convolutional Neural Networks (CNNs), a popular DL architecture for image and video analytics, can learn hierarchical representations of image features and achieve cutting-edge performance on many benchmarks. Likewise, in natural language processing, DL has achieved breakthroughs in tasks such as sentiment analysis, machine translation, and question answering. Recurrent Neural Networks (RNNs) and Transformers are common DL architectures for processing sequential and textual data. [2]

Deep learning is also being applied in many other domains such as Healthcare, Finance and Transportation. For example, DL models have been developed for medical image analysis, drug discovery, and personalized treatment planning. In finance, deep learning algorithms are used for fraud detection, credit risk assessment, and trading strategy optimization. In transportation, deep learning has been used for autonomous driving, traffic prediction, and route planning. These are just a few examples of the wide range of applications DL can enable, and the field is constantly evolving with new breakthroughs and innovations. [3]

**4. Deep neural network architectures:**

Deep Learning is a growing field with applications that span across several use cases. In each use case, a different architecture is predominant and gives the best efficiency.

There are many different types of deep learning architectures, many of which are derived from original architectures.

Some of the most popular ones are Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory Networks (LSTMs).

For the purposes of this discussion, we will talk about Convolutional Neural Networks, Recurrent Neural Networks and focus on one type of architecture known as transformers, which have gained popularity in recent years for their ability to process sequential data with parallelization and attention mechanisms.

**4.1 Convolutional Neural Networks**

**4.1.1 Introduction to Convolutional Neural Networks (CNNs)**

Convolutional Neural Networks (CNNs) are a type of neural network that are particularly well-suited for image recognition tasks. They were first introduced by Yann LeCun and his colleagues in 1998 for the purpose of recognizing handwritten digits in images. Since then, CNNs have become a popular tool for a wide range of image processing tasks, including object detection, facial recognition, and image segmentation.

CNNs are designed to mimic the way that the visual cortex processes information in the brain. They consist of a series of layers that are designed to extract progressively more complex features from an image. The first layers in the network extract simple features, such as edges and corners, while later layers extract more complex features, such as shapes and patterns.

**4.1.2 How CNNs Work**

CNNs are composed of several layers, each of which performs a specific type of computation. The most basic layers in a CNN are convolutional layers, which perform a mathematical operation called convolution. Convolution involves multiplying a small matrix of values (known as a filter or kernel) with a subset of the input image, and then summing the results to produce a single output value. The filter is then moved to the next position in the input image, and the process is repeated until the entire image has been processed.

Convolutional layers are typically followed by pooling layers, which downsample the output of the convolutional layer by taking the maximum value within a small window. This helps to reduce the dimensionality of the input, while preserving the most important features.

The final layers in a CNN are fully connected layers, which are similar to the layers in a traditional neural network. These layers take the output of the convolutional and pooling layers and use it to classify the input image.

**4.1.3 Common Architectures of CNNs**

There are several common architectures of CNNs that are used for various image processing tasks. One of the most well-known architectures is the LeNet-5 architecture, which was the first successful CNN for handwritten digit recognition. LeNet-5 consists of two convolutional layers, followed by two pooling layers, and two fully connected layers.

Another common architecture is the AlexNet architecture, which won the ImageNet Large Scale Visual Recognition Challenge in 2012. AlexNet consists of five convolutional layers, followed by three fully connected layers. One of the key innovations of AlexNet was the use of ReLU activation functions, which help to prevent the vanishing gradient problem that can occur with sigmoid activation functions.

More recently, there have been several architectures that use residual connections to help with training deep neural networks. These architectures include ResNet, DenseNet, and InceptionNet.

**4.1.4 Applications of CNNs**

Convolutional Neural Networks (CNNs) have found numerous applications in computer vision, and have achieved state-of-the-art performance in many tasks. One of the most common applications of CNNs is image classification, where the network is trained to recognize different objects or classes in an image. CNNs have also been used for object detection, which involves localizing the objects in an image and predicting their classes. This application is used in autonomous driving, robotics, and many other fields.

Another important application of CNNs is facial recognition, which involves identifying individuals from their facial features. This application has become increasingly popular in security systems, social media, and entertainment. CNNs have also been used for semantic segmentation, which involves partitioning an image into different regions and assigning a label to each region. This application is used in medical imaging, where it helps doctors to identify different structures in an image and diagnose diseases.

CNNs have also been applied to natural language processing, where they have achieved state-of-the-art performance in tasks such as sentiment analysis, text classification, and machine translation. In addition, CNNs have been used for speech recognition, where they have been combined with recurrent neural networks (RNNs) to create hybrid models that can handle both the temporal and spectral aspects of speech.

Overall, CNNs have found numerous applications in many fields, and their success is attributed to their ability to learn hierarchical and complex features from data, as well as their ability to generalize to new examples.

**4.2 Recurrent Neural Networks:**

**4.2.1 Introduction to Recurrent Neural Networks (RNNs):**

Recurrent Neural Networks (RNNs) are a type of neural network that can operate on sequences of data, making them particularly useful for time series analysis and natural language processing tasks. RNNs have the ability to store information in memory, which allows them to process sequential data and make predictions based on context. In this report, we will explore how RNNs work, their common architectures, and their applications.

**4.2.2 How RNNs Work:**

RNNs operate by processing input sequences one element at a time, and maintaining a hidden state that stores information about the sequence seen so far. The hidden state is updated at each time step, by combining the current input with the previous hidden state using a set of learned parameters. The output of the RNN is generated by applying a non-linear function to the hidden state. This process can be repeated for multiple time steps, allowing the RNN to model long-term dependencies and capture complex patterns in the input sequence.

**4.2.3 Common Architectures of RNNs:**

One of the most popular RNN architectures is the Long Short-Term Memory (LSTM) network. LSTMs were designed to address the vanishing gradient problem that can occur in traditional RNNs, which can make it difficult for the network to learn long-term dependencies. LSTMs achieve this by using a set of gating mechanisms that regulate the flow of information through the network, allowing it to selectively remember or forget information from previous time steps.

Another popular architecture is the Gated Recurrent Unit (GRU) network, which is similar to the LSTM but has fewer gating mechanisms. This makes the GRU simpler to train and faster to compute than the LSTM, while still achieving good performance on many tasks.

**4.2.4 Applications of RNNs:**

RNNs have found numerous applications in natural language processing, including language modeling, text classification, and machine translation. They have also been used for speech recognition, where they can model the temporal dynamics of speech signals. In addition, RNNs have been used for time series forecasting, where they can predict future values based on past observations.

RNNs have also been applied to music generation, where they can learn to produce new musical compositions based on existing music. They have also been used for video analysis, where they can model the temporal dynamics of video sequences and perform tasks such as action recognition and video captioning.

Overall, RNNs have proved to be a powerful tool for processing sequential data and have found numerous applications in many fields. Their ability to model long-term dependencies and capture complex patterns in sequential data makes them an essential tool for many machine learning tasks.

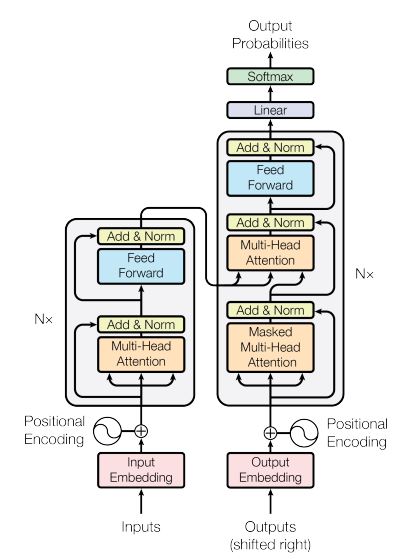
**4.3 Transformers:**

**4.3.1 Introduction to Transformers in Deep Learning:**

Recurrent neural networks (RNNs) have been the go-to model for sequence processing tasks in deep learning. However, RNNs have several limitations, including difficulty in parallelization and difficulty in capturing long-term dependencies. These limitations have led to the development of alternative models, such as transformers, which have become increasingly popular in recent years. Transformers are deep learning models that analyze input and output data, primarily used in natural language processing (NLP) and computer vision (CV). They are neural networks that learn context and understanding through sequential data analysis, using a modern and evolving mathematical technique set known as attention or self-attention.

**4.3.2 The Transformer Architecture: key components and functionality:**

Most competing neural sequence transduction models have an encoder-decoder structure. Here the encoder maps a sequence of input symbols represent (x1, ..., xn) to a sequence of consecutive representations z = (z1, ..., zn). Given z, the decoder then produces an output sequence (y1,...,ym) Symbols one item at a time. At each step the model is autoregressive, Generate with previously generated symbols as additional input Next. Transformer is passed as encoder and the decoders are shown in the left and right halves of Figure 1, respectively.[4]



**4.3.2.1 Encoder and Decoder Stacks :**

**Encoder:** The encoder consists of a stack of N = 6 identical layers. Each layer has two sublayers. The first is a multi-head self-aware mechanism, and the second is a simple, position-aware, fully-connected feed-forward network. The architects used residual connections around each of the two sublayers, followed by layer normalization. That is, the output of each sublayer is LayerNorm(x + Sublayer(x)), where Sublayer(x) is a function implemented by the sublayer itself. To facilitate these residual connections, all sublayers in the model as well as embedding layers produce outputs of dimension dmodel = 512.[4]

**Decoder:** The decoder also consists of a stack of N = 6 identical layers. In addition to the two sublayers in each encoding layer, the decoder inserts a third sublayer that performs multi-head attention on the output of the encoding stack. Similar to the encoder, the architecture uses residual connections around each sublayer followed by layer normalization. It also modifies the self-awareness sublayer in the decoder stack to prevent positions from obeying subsequent positions. This masking, combined with the fact that the output embedding is offset by one position, ensures that a prediction at position i can only depend on known outputs at positions smaller than i. [4]

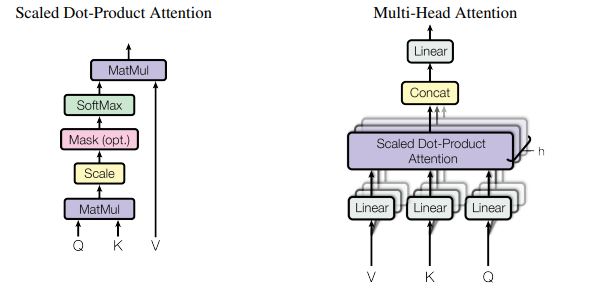
**4.3.2.2 Attention :**

An attention function can be described as mapping a query and a set of key-value pairs to an output, where query, key, value, and output are all vectors. The output is computed as a weighted sum of values, where the weight assigned to each value is computed by the query compatibility function with the appropriate key.

**4.3.2.2.1 Scaled Dot-Product Attention:**

The input consists of queries and keys of dimension dk, and values of dimension dv. Here they compute the dot products of the query with all keys, divide each by √ dk, and apply a softmax function to obtain the weights on the values.

**4.3.2.2.2 Multi-Head Attention:**

Instead of performing a single attention function with dmodel-dimensional keys, values and queries, multi-head attention linearly project the queries, keys and values h times with different, learned linear projections to dk, dk and dv dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding dvdimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2. [4]

**4.3.2.2.3 Applications of attention in transformer model:**

The Transformer uses multi-head attention in a different way: in the "encoder-decoder attention" layer, the query comes from the previous decoder layer, and the storage keys and values ​​come from the output of the encoder. This allows any position in the decoder to have all positions in the input sequence. This mimics the typical encoder-decoder attention mechanism in sequence-to-sequence models.

Encoders contain multiple layers of self-awareness. In the self-aware layer, all keys, values, and queries come from the same place, in this case the output of the previous layer in the encoder. Every position in the encoder can serve all positions in the layer above the encoder.

**4.3.2.3 Position-wise Feed-Forward Networks :**

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between. FFN(x) = max(0, xW1 + b1)W2 +b2

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is dmodel = 512, and the inner-layer has dimensionality df f = 2048. [4]

**4.3.2.4 Embeddings and Softmax:**

Like other models that deal with sequence transduction, the Transformer architecture utilizes embeddings that are learned to convert the input and output tokens into dmodel-dimensional vectors. Additionally, the Transformer uses standard learned linear transformations and a softmax function to generate predicted probabilities for the next token based on the decoder output. The embedding layers of the Transformer model share the same weight matrix with the pre-softmax linear transformation layer, which is similar to other models. To scale the weights in the embedding layers, the Transformer multiplies them by the square root of dmodel. [4]

**4.3.2.5 Positional Encoding:**

As the model lacks both recurrence and convolution, the Transformer must incorporatec information about the order of the sequence in order to make use of it. To accomplish this, positional encodings are added to the input embeddings at the bottom of the encoder and decoder stacks. These encodings convey information about the relative or absolute position of the tokens in the sequence. The positional encodings have the same dimension as the embeddings (dmodel), allowing the two to be summed. The Transformer architecture offers a variety of choices for the positional encodings, which can either be learned or fixed. [4]

**4.3.3 Advantages of Transformers:**

Parallelization: Transformers are designed to process input data in parallel, allowing for efficient use of resources such as GPUs and TPUs. This results in faster training and inference times compared to traditional recurrent neural network (RNN) models.

Attention Mechanism: The attention mechanism in Transformers allows the model to focus on the most important parts of the input sequence, while ignoring irrelevant information. This results in better performance on tasks such as language modeling, machine translation, and text classification.

Pre-training: The Transformer architecture can be pre-trained on large amounts of text data using unsupervised learning methods, such as the masked language modeling task used in BERT. This pre-training can then be fine-tuned for specific downstream NLP tasks, resulting in better performance with smaller amounts of task-specific training data.

Transfer Learning: Pre-trained Transformer models can be easily adapted to new NLP tasks by fine-tuning the existing model on task-specific data. This reduces the need for large amounts of task-specific data and allows for transfer learning across different NLP tasks. Long-term Dependencies: Transformers are designed to handle long-term dependencies in input sequences, which is important for NLP tasks such as language modeling and machine translation where understanding the context of a sentence or document is crucial.[4]

**5.** **Conclusion:**

In this chapter we have presented the important notions that are related to deep learning (definition, Architectures….etc). As well as a general vision on deep learning, all giving in detail the method chosen in our research work which is the "Transformers".

**References:**

[1] Goodfellow, Ian, et al. "Deep Learning." MIT Press, 2016.

[2] LeCun, Yann, et al. "Deep Learning." May 2015 Nature 521(7553):436-44

[3] Bengio, Yoshua, et al. "Deep Learning." Nature, vol. 521, no. 7553, 2015.

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All You Need. In Advances in Neural Information Processing Systems (NIPS), pp. 5998-6008.