# CS546 Project Report: Automatic Generation of EEG Report Using Transformers

Jialong Yin, Peiyao Li, Kaichen Le, and Beichen Zhang

University of Illinois at Urbana-Champaign, Champaign IL 61820, USA
{jialong2, peiyaol2, kle11, bzhang64}@illinois.edu

**Abstract.** Electroencephalogram (EEG) is widely used for medical professionals to identify possible brain disorders. However, manual EEG report creation can be time-consuming and error-prone. To combat these issues, we proposed an automatic approach to generate EEG reports using Transformers. We named our model EEG Transformers, and it consists of three parts: the EEG encoder, the Transformer encoder, and the Transformer decoder. Although our model has not achieved state-of-the-art performance, it still produces promising results. All of the works described in this report are done specifically for this class and will be extended into the future.

**Keywords:** Medical Text Generation · Electroencephalogram (EEG) · Transformers.

## 1 Introduction

Electroencephalogram (EEG) is oftentimes used to detect brain activities by measuring the electrical impulses of the brain [7]. EEG can be very useful in identifying and diagnosing brain disorders, such as seizures and epilepsy [7]. However, the traditional way of diagnosing patients though EEG involves human inspection and manual creation of reports. Just like the traditional way of generating other medical reports such as radiology reports [5], this requires time to train medical professionals as well as time to write the reports [1]. Manual report writing also creates more room for human error [1, 5].

To combat the issues in manual reporting, we propose a neural network model based on Transformers to facilitate automatic EEG report generation. Although there are previous works done on automatic EEG report generation [1], there has not been an approach that utilizes Transformers. Transformer [13] is a neural network model that becomes widely effective in pure NLP tasks first and is later applied to many other extensions such as video captioning [18] and bioinformatics [11]. It can capture long-range dependencies in sequences more easily and facilitates the training process with more parallelization. In our work, we apply Transformers to generate EEG reports automatically.

We propose a model called EEG Transformer, which contains three components: EEG encoder, Transformer encoder, and Transformer decoder. The EEG encoder transforms the EEG signals into feature vectors through temporal convolutional layers. The Transformer encoder uses self-attention mechanisms to transform the EEG feature vectors into dense representations. Finally, the Transformer decoder takes the output from the Transformer encoder and the preprocessed report vectors. It outputs the probability of words in the auto-generated report.

For the experiments, we implement the code in Pytorch and use Bluewater as the computational resource. We use the TUH dataset from Temple University [10] and preprocess both the reports and the EEG signals. We compare our results to EEGtoText, as well as several other baselines presented by Biswal et al. [1] We use both METEOR and BLEU as the evaluation metrics. Although our model has not reached state-of-the-art performance compared to EEGtoText, it still shows promising results given that we have more limits on the amount of data to train. In the future, we plan to improve by using the entire dataset to train.

## 2 Related Works

### 2.1 Automatic Medical Report Generation

**X-ray** Past works have been done on automatic medical report generation. Particularly, most works have a focus on radiology images. For example, Shin et al. propose a neural network model to automatically detect diseases from chest x-rays [4]. Shin et al. use a combination of CNNs and RNNs: first, CNN models are trained with x-ray images and one disease label each; then, RNNs are used to infer the context of the disease labels; finally, CNN models are retrained given the image and the contexts [4]. Similarly, Wang et al. propose a model named TieNet, which combines multi-levels attention models with end-to-end trainable CNN-RNN architecture [15]. Later, Jing et al. propose another approach to automatically generate x-ray reports [5]. Jing et al. use a hierarchical LSTM, combined with co-attention networks that take in both visual and semantic features to generate the topic of the report [5]. More recently, Xie et al. extend on LSTM-based methods by adding a topic guide attention module that can switch between normal context and abnormal context [16].

**EEG** Although most past works on automatic medical report generation have been on x-rays, there are a few works that focused on EEG report generation. One of the most significant differences between EEGs and x-rays is that unlike the x-rays which are static, EEGs are temporal series of variable length input. Biswal et al.'s EEGtoText is one method that automatically generates medical reports from EEGs [1]. Our work is also based on EEGtoText, and we originally want to use it as the baseline. EEGtoText uses a hierarchical LSTM that contains both paragraph LSTMs and sentence LSTMs [1]. EEGtoText also categorizes EEG/report combination into different phenotypes and uses different templates according to the phenotype [1]. There are a few shortcomings of EEGtoText, particularly it has a slow training process and it is not end-to-end training. Since we don't have the capacity to improve on the phenotype templates, we focus on improving the model using Transformers instead of LSTMs.

### 2.2 Image and Video Captioning

**Image Captioning** Image captioning is one task that has quite a few similarities to medical report generation. Vinyals et al. propose a very influential model which is the CNN-RNN combination [14]. Vinyals et al. use CNN to generate image embeddings and feed into a sequence of LSTMs to generate the probability of words at each position [14]. A lot of the later works are built upon Vinyals et al.'s CNN-RNN architecture, for example, Pedersoli et al. extend the CNN-RNN architecture by adding "Areas of Attention", where the generation of each word is also dependent on specific regions of the images [9]. Wang et al. also improved on the CNN-RNN architecture by using bi-directional LSTMs to improve the accuracy in caption generation [3].

**Video Captioning** Compared to image captioning, video captioning is more related to EEG report generation because they both have variable-length input. Some works have developed models for video captioning based on the CNN-RNN structure for image captioning. For example, to capture video embeddings, Yu et al. use an attention model that aligns words at specific timestamps with specific parts of the video, while keeping the basic CNN-RNN structure [17]. Yu et al. also use hierarchical GRUs for sentence generation and paragraph generation [17]. Similarly, Li et al. develop a multitask reinforcement learning model that uses multiple copies of CNNs to encode the video at different timestamps [6]. However, more recent works have shifted the focus to Transformers. Zhou et al. propose a model that uses masked Transformers with temporal convolutional networks and self-attention mechanisms to precisely determine the topic for captioning at specific timeframes [18]. Since there has been a success at using Transformers for the task of video captioning, we believe that we can also apply Transformers to generate EEG reports.

## 3 Model

### 3.1 Preliminary

Our model is built on the Transformer [13] model proposed by Ashish, Noam, Niki, etc. in their paper called 'Attention Is All You Need'. In this section, we are going to briefly introduce the Transformer and discuss the reason why it is chosen as the base model we built on.

Transformer is a transduction model that consists of an encoder-decoder structure and uses multi-headed self-attention. One novel composition of the Transformer model is the self-attention mechanism. With query, key, and information to be extracted as input, it can calculate the scaled dot-product attention [13]. Multiple independent scaled dot-product attention can form the multi-head attention where different heads, the dot-product attention layers, help to capture different aspects of information. Because the self-attention does not hold any position information, to capture the position information, the transformer adds position vectors into the embeddings wherein the paper each position has a unique, manually set position vector. For the encoder-decoder structure of the Transformer, the encoder has a self-attention layer to take in embeddings and outputs the output to the decoder after a fully connected feed-forward layer. The decoder consists of two same layers with the same size but an additional multi-head attention layer to process the output from the encoder.

We chose the Transformer to be our building block because of its speed and its accuracy on processing sequential data contrasts to other proposed models. The recurrent neural network can hardly speed up the job because the structure limits the job from running in parallel. The convolutional neural network requires a high stack of layers to consider the context over a long sequence which has large limitations on accuracy when context-dependency is an important factor. With the self-attention mechanism, the Transformer solves both of the problems which make it an ideal model to work on.

### 3.2 Data and Task Definition

The task of our model is to take Electroencephalography(EEG) data as input and generates corresponding diagnoses.

EEG is a medical test that monitors people's brain activity by recording brain waves, the electrical impulses that brain cells communicate with each other, using electrophysiological methods [2]. In real life, patients who have brain injury will have the EEG test. And then, doctors will give diagnoses based on the EEG diagrams. EEG can have various channels based on the number of electrodes placed on the head [8]. This means EEG recording can have various numbers of channels and each channel demonstrates different parts of the brain activity. Moreover, except for the number of channels, the length of each EEG recording varies based on the time of the test. This means EEG recording is a sequential data that has various lengths.

The task of the EEGTransformer is to construct a transformer that reads in EEG records as input and generates the most possible clinics report containing several paragraphs.

### 3.3 EEG Transformer Framework

As drawn in Figure 1, the EEG Transformer model is formed by three components, EEG encoding, Transformer encoder, and Transformer decoder. EEG encoding transfers the EEG record vector to feature vectors and passes the feature vectors to the Transformer. The transformer encoder encodes the feature vectors into a dense presentation. And then the Transformer decoder decodes the presentation with the diagnosis token to generate the clinic report.

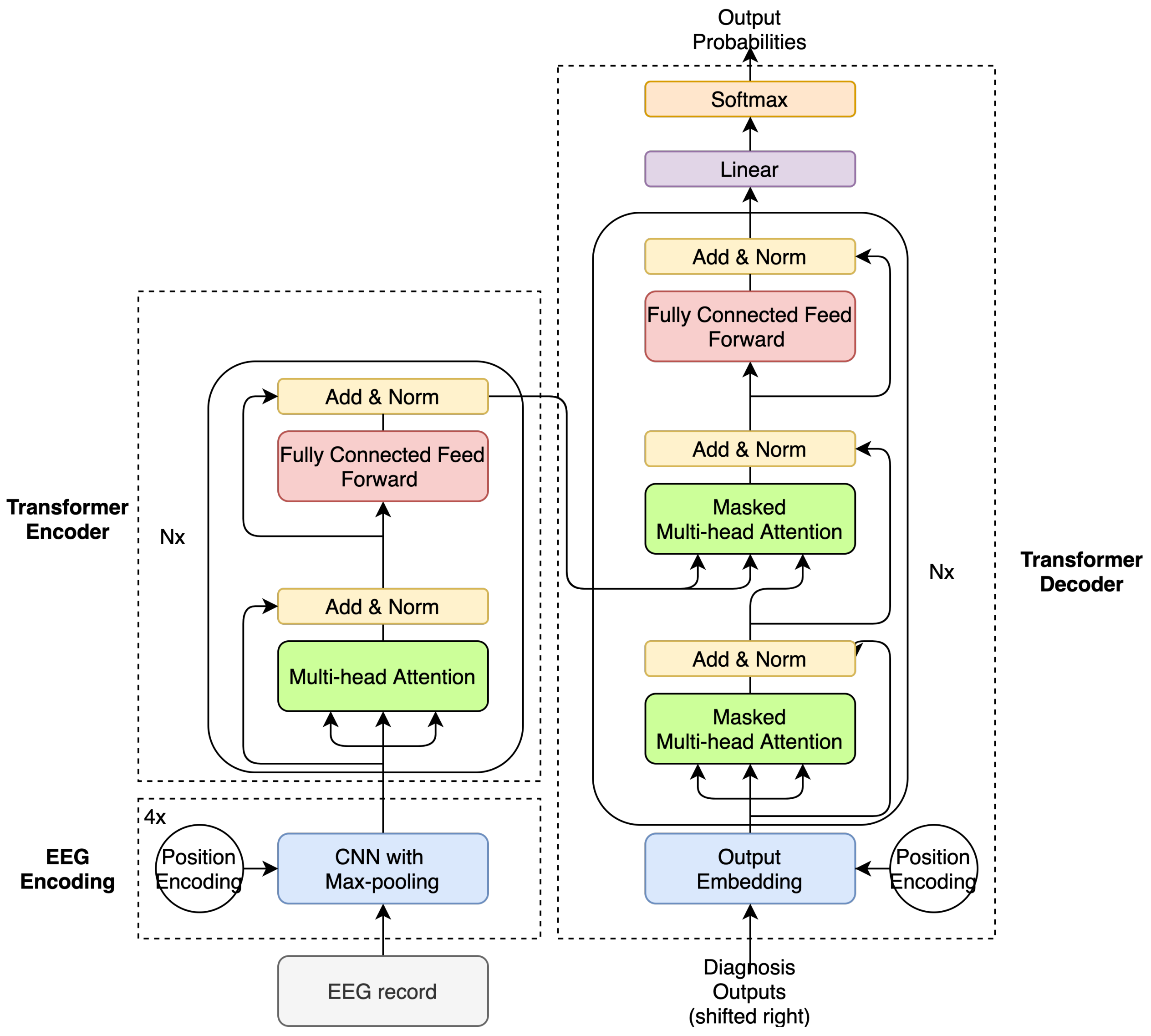

**Fig. 1.** EEG Transformer framework

**EEG Encoding** EEG encoding transfers the EEG record into feature vectors through a CNN with convolutional max-pooling just same as the method that Siddharth used in his paper ‘EEGtoText: Learning to Write Medical Reports from EEG Recordings’ [1]. In more detail, we preprocess the EEG record to EEG epochs of one minute as X = $x_1, x_2, x_3, ..., x_T$. And then, we pass each $x_i$ to the CNN and get the corresponding feature vector $f_i$. That is

$$f_i = CNN(x_i),\ where\, i \in [1,T]\ and\, T\, is\, the\, number\, of\, discretized\, time\, steps\, per\, recording$$

Rectified Linear Units(ReLUs) activation function and batch normalization are also used in the CNN [1]. The whole process is repeated four times to achieve the final feature vectors.

**Transformer Encoder** The Transformer encoder takes EEG feature vectors from the EEG encoding as input and outputs the dense representation to the Transformer decoder by doing self-attentions. Because the self-attention mechanism does not consider the position information, position encoding is done by add manually set position vector to the input embeddings just as stated in the original paper [13]. Because each epoch have a different frequency which results in various length of input and each of the input is null-filled to have the same length, source key padding mask and memory key padding mask are implemented to deal with the various length of input embedding.

**Transformer Decoder** The Transformer decoder takes the input of the dense representation of EEG feature vectors from the Transformer encoder, as well as the pre-processed target report vectors, and output the probability of the diagnosis report. During the implementation, similar position coding is done on the target. The target mask is used to prevent the decoder see the future tokens that have not been processed. Target key padding mask which is similar to the source key padding mask is used for ignoring the null-filled part of the vectors because the clinical report also has various length and the empty spots are filled with null.

### 3.4 Training and Inference

The loss function l(output, target) for training combines a softmax function followed by a logarithm, and the negative log-likelihood loss by using the CrossEntropyLoss function from PyTorch library [12].

$$\operatorname{loss}(output,\ \text{target}) = -\log\left(\frac{\exp(output[\text{ target }])}{\sum_j \exp(output[j])}\right) = -output[\text{ target }] + \log\left(\sum_j \exp(output[j])\right)$$

[12] For inference, we use the report generated by the Transformer decoder using dense feature vectors from the Transformer encoder.

## 4 Experiments and Results

For the experimental part, we would like to present some details about the TUH dataset we used, our architecture implementation, insights about the evaluation strategy and their result comparing with the available baseline like: S2VT, Temporal Attention Network, Softmax and Mean-Pooling which we are going to introduce in the later sections.

### 4.1 Datasets and Preprocessing

For the dataset we are using the TUH dataset from Temple University [10]. The TUH dataset is the most comprehensive dataset available for Electroencephalography resource. It contains data of 10,865 patients alone with 16,950 Electroencephalography data sample and their corresponding reports and 542,765 data tokens that formed 3,452 hours long Electroencephalography data length and 542,765 tokens. The dataset is huge as each eeg data sample consists of signal data for each electrode labeling in 10–20 system per labeling per frequency. A single eeg sampel alone takes about 20 megabytes. Under the hazard COVID 19 pandemic condition, we do not have enough computing resources to run the full dataset in TUH. Resulting from that, data preprocessing and data downsampling become extremely important to our performance of the model. The dataset is not only huge but also comprehensive, in report data, it includes clinic history, medicine, introduction, impression, description of the record and clinical correlation. For our experiment, we only use impression and description of the record for supervised learning. But we do not exclude the possibility of generating other part of the report in the future research.

The most important thing we need do is to downsample the channel. 10-20 system obtain its channel by subtracting one electrode labeling with another. A comprehensive analysis for EEG suppose to have 18 channels in total. We use the four major channels to meet our computing requirement. The channels we used are 'FP1 - F3', 'F3 - C3', 'C3 - P3', 'P3 - O1' where FP1, F3, C3, P3 and O1 are all electrode labeling. When dealing with the sign data, we realize that the EEG data in the TUH dataset have inconsistent frequency, which will result in inconsistent signal time length when modeling. Thus we preprocess the frequency using interpolation. For the report data, we keep track of the word bag and preprocess the text to remove some noise data.

## 4.2 Experimental Setup

For the training hyperparameter, we are using a learning rate of 0.001, batch size of 32, epoch of 50 and we do evaluation every 10 epoches. We implement the model in pytorch and we train the model on BlueWater. We split 10 percent data for the validation set.

**Baselines** There are many models available for us to do comparison. We are mainly comparing our model with EEGtoText which share same task objective as us without using the transformer. In the meantime we are also comparing with S2VT, Temporal Attention Network, Softmax and Mean-Pooling.

1. EEGtoText: Using stacked CNN and R-CNN to extract patterns and doing classification to get key phenotypes, EEGtoText can then generate the impression section. Then they use paragraph-level and sentence-level Long-short-term Memory to generate detailed description. The model reach state of the art performance. (Biswal et al., 2019)
2. S2VT: encoding using CNN and applying stacked LSTM to build a seq2seq model to generate EEG medical report.(Venugopalan et al., 2015)
3. Temporal Attention Network: generate medical report using Encoder Decoder architecture that use CNN as encoder and Temporal Attention as decoder. (Venugopalan et al., 2015)
4. Soft Attention: Using soft attention for decoder to focus more on feature. (Venugopalan et al., 2014)
5. Mean-Pooling: Similar model with S2VT but use Mean-Pooling to process hidden layer. (Venugopalan et al., 2014)

**Architecture Details** We use architecture table below. Layer 1-15 are encoder where we use Conv1d, MaxPool1d and BatchNorm1d where we encode, downsample and normalize the eeg data. Layer 16 is the embedding look up table. Layer 17-18 are positional encoding where we put relative position in text sequence into the layers. Layer 19-24 and layer 25-29 are two transformer we use for decoding. And last but not least, layer 30-32 are word-net where we match the output to text.

| | | **Model Architecture** | | |
|---|---|---|---|---|
| *Layer No.* | *Layer Type* | *Key Parameter* | *Input Channels* | *Output Channels* |
| 1 | Conv1d | 5 | 4 | 32 |
| 2 | MaxPool1d | 14 | 32 | 32 |
| 3 | BatchNorm1d | 32 | 32 | 32 |
| 4 | Dropout | 0.15 | 32 | 32 |
| 5 | Conv1d | 5 | 32 | 64 |
| 6 | MaxPool1d | 3 | 64 | 64 |
| 7 | BatchNorm1d | 64 | 64 | 64 |
| 8 | Dropout | 0.15 | 64 | 64 |
| 9 | Conv1d | 5 | 64 | 256 |
| 10 | MaxPool1d | 6 | 256 | 256 |
| 11 | BatchNorm1d | 256 | 256 | 256 |
| 12 | Dropout | 0.15 | 256 | 256 |
| 13 | Conv1d | 5 | 256 | 512 |
| 14 | MaxPool1d | 7 | 512 | 512 |
| 15 | BatchNorm1d | 512 | 512 | 512 |
| 16 | Embedding | 60 | 512 | 512 |
| 17 | Dropout(eeg) | 0.1 | 512 | 512 |
| 18 | Dropout(report) | 0.1 | 512 | 512 |
| 19 | Linear | True | 512 | 256 |
| 20 | Dropout | 0.1 | 256 | 256 |
| 21 | Linear | True | 256 | 512 |
| 22 | LayerNorm | (512,) | - | - |
| 23 | Dropout | 0.1 | 512 | 512 |
| 24 | LayerNorm | (512,) | - | - |
| 25 | Linear | True | 512 | 256 |
| 26 | Dropout | 0.1 | 256 | 256 |
| 27 | Linear | True | 256 | 512 |
| 28 | LayerNorm | (512,) | - | - |
| 29 | Dropout | 0.1 | 512 | 512 |
| 30 | Linear | True | 512 | 512 |
| 31 | Tanh | - | - | - |
| 32 | Linear | True | 512 | 1414 |

**Evaluation Metrics** For image and video caption related task, the most popular evaluation methods are BLEU (bilingual evaluation understudy), Metric for Evaluation of Translation with Explicit Ordering (METEOR) and Consensus-based Image De- scription Evaluation(CIDEr). These algorithms can effectively measure the similarity between the experiment output and given label (report in the dataset).

- METEOR is short for Metric for Evaluation of Translation with Explicit Ordering. It find the alignment from output to target with minimum intersection. It then computes the precision and recall and get Fmean score based on Precision and Recall. (Banerjee and Lavie, 2005)
- BLEU is short for Bilingual Evaluation Understudy Score. The algorithm is based on n-gram precision score with weight over the sentence length. Short sentence get penalized. (Papineni et al., 2002)

### 4.3 Results and Analysis

The result is shown below:

| Method | Meteor | BLEU 1 | BLEU 2 | BLEU 3 | BLEU 4 |
|---|---|---|---|---|---|
| MP | 0.345 | 0.645 | 0.578 | 0.459 | 0.361 |
| S2VT | 0.361 | 0.724 | 0.613 | 0.543 | 0.438 |
| TAM | 0.369 | 0.714 | 0.647 | 0.492 | 0.461 |
| SA | 0.353 | 0.736 | 0.619 | 0.519 | 0.420 |
| EEGtoText | 0.358 | 0.758 | 0.639 | 0.611 | 0.483 |
| Our Model | 0.050 | 0.561 | 0.263 | 0.136 | 0.058 |

The results from our model do not reach state of the art performance due to limited memory size of computing resources during training. As we can see there are significant performance differences with EEGtoText model. The performance however, is under our expectation since the either the training data size or the epoch size are large enough to be reach the optimal performance. We can still see quite impressive BLEU score when n in n-gram is small. It means the model can capture local information while the global information remain unsatisfied. We expect to see improvement in evaluation score once we get access to more computational resources.

## 5 Conclusion

We proposed an encoder decoder architecture based Electroencephalogram medical generation model. It takes EEG signal data as input and output the impression and description part of the report. We use traditional stacked CNN and positional encoding for encoder and we use innovative transformer with word-net to be the decoder. The model reach reasonable performance under METEOR and BLEU measurement considering the size of the training data. In the future we plan to train our model in full dataset and expand its functionality to predict other sections of the report.

## References

1. Biswal, S., Xiao, C., Westover, M.B. & Sun, J.. (2019). EEGtoText: Learning to Write Medical Reports from EEG Recordings. Proceedings of the 4th Machine Learning for Healthcare Conference, in PMLR 106:513-531.
2. Blocka, K. (2018, September 29). EEG (Electroencephalogram): Purpose, Procedure, and Risks. Retrieved May 15, 2020, from https://www.healthline.com/health/eeg
3. Cheng Wang, Haojin Yang, Christian Bartz, and Christoph Meinel. 2016. Image Captioning with Deep Bidirectional LSTMs. In Proceedings of the 24th ACM international conference on Multimedia (MM '16). Association for Computing Machinery, New York, NY, USA, 988–997. DOI:https://doi.org/10.1145/2964284.2964299.
4. H. Shin, K. Roberts, L. Lu, D. Demner-Fushman, J. Yao and R. M. Summers, "Learning to Read Chest X-Rays: Recurrent Neural Cascade Model for Automated Image Annotation," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2497-2506, doi: 10.1109/CVPR.2016.274.
5. Jing, Baoyu, Pengtao Xie, and Eric Xing. "On the Automatic Generation of Medical Imaging Reports." Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2018): n. pag. Crossref. Web.
6. Li, Lijun & Gong, Boqing. (2018). End-to-End Video Captioning with Multitask Reinforcement Learning. https://arxiv.org/abs/1803.07950.
7. Mayo Clinic. EEG (electroencephalogram). https://www.mayoclinic.org/tests-procedures/eeg/about/pac-20393875. Last accessed May 14 2020.
8. Multi-Channel EEG (BCI) Devices. (2015, July 7). Retrieved May 15, 2020, from http://neurosky.com/2015/07/multi-channel-eeg-bci-devices/
9. M. Pedersoli, T. Lucas, C. Schmid and J. Verbeek, "Areas of Attention for Image Captioning," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 1251-1259, doi: 10.1109/ICCV.2017.140.
10. Obeid, I., & Picone, J. (2016). The Temple University Hospital EEG Data Corpus. Frontiers in neuroscience, 10, 196. https://doi.org/10.3389/fnins.2016.00196
11. Rives, Alexander, Siddharth Goyal, Joshua Meier, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. bioRxiv:622803, 2019. doi: https://doi.org/10.1101/622803.
12. torch.nn¶. (n.d.). Retrieved May 15, 2020, from https://pytorch.org/docs/stable/nn.html
13. Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin. Attention Is All You Need. arXiv:1706.03762 [cs], 2017. http://arxiv.org/abs/1706.03762.
14. Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3156-3164. https://arxiv.org/abs/1411.4555.
15. Wang, Xiaosong, Yifan Peng, Le Lu, Zhiyong Lu and Ronald Summers. (2018). TieNet: Text-Image Embedding Network for Common Thorax Disease Classification and Reporting in Chest X-Rays. IEEE CVPR 2018. 10.1109/CVPR.2018.00943.
16. Xie X., Xiong Y., Yu P.S., Li K., Zhang S., Zhu Y. (2019) Attention-Based Abnormal-Aware Fusion Network for Radiology Report Generation. In: Li G., Yang J., Gama J., Natwichai J., Tong Y. (eds) Database Systems for Advanced Applications. DASFAA 2019. Lecture Notes in Computer Science, vol 11448. https://doi.org/10.1007/978-3-030-18590-9_64.
17. Yu, Haonan & Wang, Jiang & Huang, Zhiheng & Yang, Yi & Xu, Wei. (2015). Video Paragraph Captioning Using Hierarchical Recurrent Neural Networks. https://arxiv.org/abs/1510.07712.
18. Zhou, Luowei, Yingbo Zhou, Jason J. Corso, Richard Socher and Caiming Xiong. End-to-End Dense Video Captioning with Masked Transformer. arXiv:1804.00819 [cs.CV], 2018. https://arxiv.org/abs/1804.00819.