**Transformers Architectures**

The transformer model is a machine learning model that has its origins in natural language processing (NLP) and is used to handle sequential input data. This data is typically a sequence of word embeddings. Transformers use a mechanism called attention to relate the sequential input to each other. As it comes from the NLP subfield of machine learning, it was first introduced as a way to relate words in translations from different words to one another. Thus, capturing what data points in the input had the largest effect on the output, i.e., what to pay attention to in a sense. When the concept is applied to machine vision the implementation is changed slightly, but the intuition behind it remains useful for understanding it. In machine vision applications it captures the relationship between pixels that are part of the same object. Specifically, transformers use multi-headed attention which applies the attention mechanism several times in parallel capturing different facets of the information and how different patches relate to one another.

* **Vision Transformer (ViT)**

One of the first uses of the attention mechanism within the machine vision field was the Vision Transformer (ViT). It **splits up an image into patches of pixels**, grouped together in squares. Originally the patches were 16 by 16 pixels in size. These patches are embedded into a learned latent dimension and in turn, used analogously as word embeddings in the transformer models from the NLP field.

Dosovitskiy et al. (2021) summed up the essence of the vision transformer architecture they designed in the title of their paper: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

An image can be converted into patches of 16x16 words.

* **Swin Transformer**

Shifted windows (Swin) Transformer is a transformer architecture developed by Microsoft and **used for vision tasks**. It was first proposed in March 2021 and established a new state of the art for several vision-related tasks, among them semantic segmentation where it achieved the highest metric on the ADE20K dataset. The Swin Transformer is a hierarchical transformer architecture which **uses image patches of varying size in hierarchical layers** where the **goal is to capture both global and local features**.

The **model takes in an image which is split into several windows and processed by the first layer**. The results of this is sent to the next layer which splits it up further. Since the window size between the layers vary while the number of patches in each window is constant, the patches get merged to fit inside the windows. In this architecture the layers with smaller window sizes are used to capture smaller features while the layers with larger window sizes are used to capture more global features. In the Swin Transformer architecture, attention is a key component and is computed inside each window. To propagate information between windows they are shifted so that they overlap. This is so that pixels that are part of the same object, but not inside the same window share information.

The **use of windows in the image representation is the biggest difference between the Swin Transformer architecture and the ViT architecture is only split up into patches and not into windows of varying size**.

* **CLIP**

Contrastive Language-Image Pre-Training (CLIP) follows the **philosophy of transformers**. It plugs **sequences of data** in its transformer-type layers. Instead of sending text pairs, this time, the **model sends text-image pairs**. Once the data is tokenized, encoded, and embedded, CLIP, a task-agnostic model, learns text-image pairs as with any other sequence of data.

The method is contrastive because it looks for the contrasts in the features of the image. It is the method we use in some magazine games in which we have to find the differences, the contrasts, between two images.

* **DALL-E**

DALL-E, as with CLIP, is a task-agnostic model. **CLIP processed text-image pairs**. DALL-E processes the text and image tokens differently. DALL-E’s input is a single stream of text and image of 1,280 tokens. 256 tokens are for the text, and 1,024 tokens are used for the image. DALL-E is a foundation model like CLIP.

DALL-E was named after Salvador Dali and Pixar’s WALL-E. The usage of **DALL-E is to enter a text prompt and produce an image.** However, DALL-E must first learn how to generate images with text.

DALL-E is a 12-billion-parameter version of GPT-3.

**This transformer generates images from text descriptions using a dataset of text-image pairs**.

* **T5 Model**

Raffel et al. (2019) designed a transformer meta-model based on a simple assertion: every NLP problem can be represented as a text-to-text function. Every type of NLP task requires some kind of text context that generates some form of text response.

A text-to-text representation of any NLP task provides a unique framework to analyze a transformer’s methodology and practice. The idea is for a transformer to learn a language through transfer learning during the training and fine-tuning phases with a **text-to-text approach**.

Raffel et al. (2019) named this approach a **T**ext-**T**o-**T**ext **T**ransfer **T**ransformer. The 5 Ts became T5, and a new model was born. Raffel et al. (2019) designed a conceptual text-to-text model and then implemented it. Raffel et al. (2019) proposed to add a prefix to an input sequence which solved the problem of unifying task-specific formats & that way the model parameters would be trained for all types of tasks with one text-to-text format. The idea was to find a way to have one input format for every task submitted to the transformer.

The core concept of a T5 model is to find an abstract model that can do things like us.

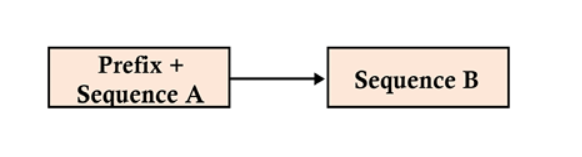


Figure-1: Unifying the input format of a transformer model

The unified input format leads to a transformer model that produces a result sequence no matter which problem it has to solve in the T5. The input and output of many NLP tasks have been unified:

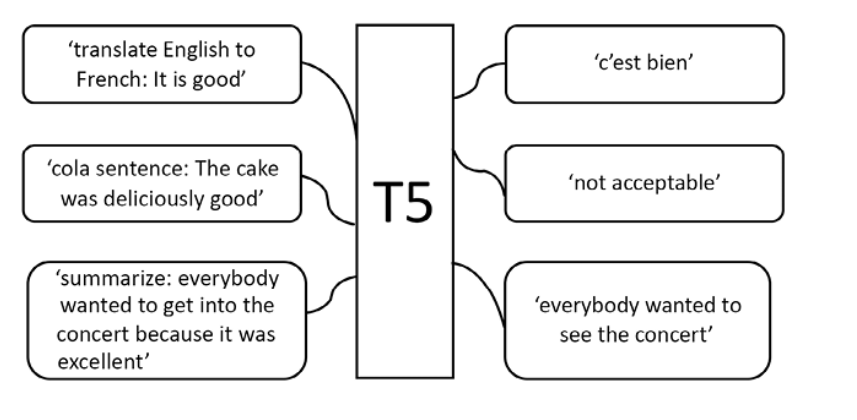


Figure-2: The T5 text-to-text framework

The unification process makes it possible to use the same model, hyperparameters, and optimizer for a wide range of tasks.

**DistilBERT, ELECTRA, and RoBERTa**

**DeBERTa**

Another new approach to transformers can be found through **disentanglement**. Disentanglement in AI allows you to separate the representation features to make the training process more flexible.

The two main ideas implemented in DeBERTa are:

* Disentangle the content and position in the transformer model to train the two vectors separately
* Use an absolute position in the decoder to predict masked tokens in the pretraining process