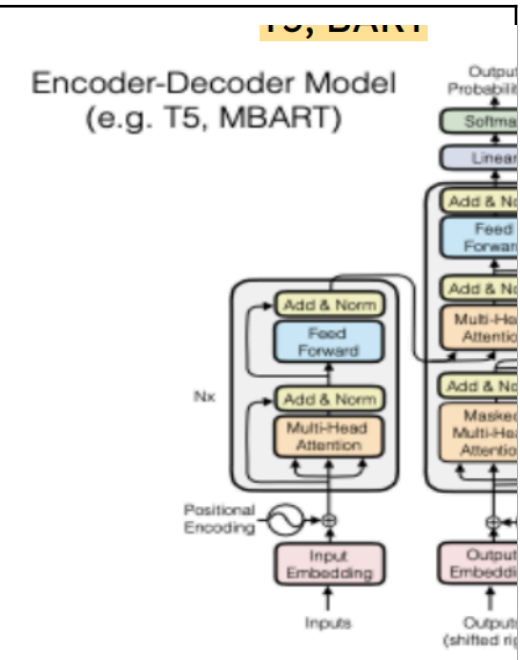
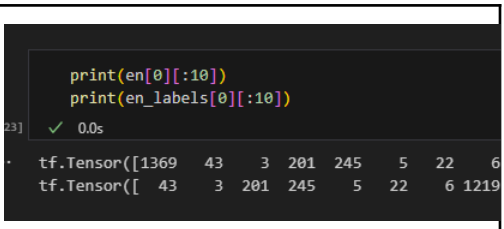
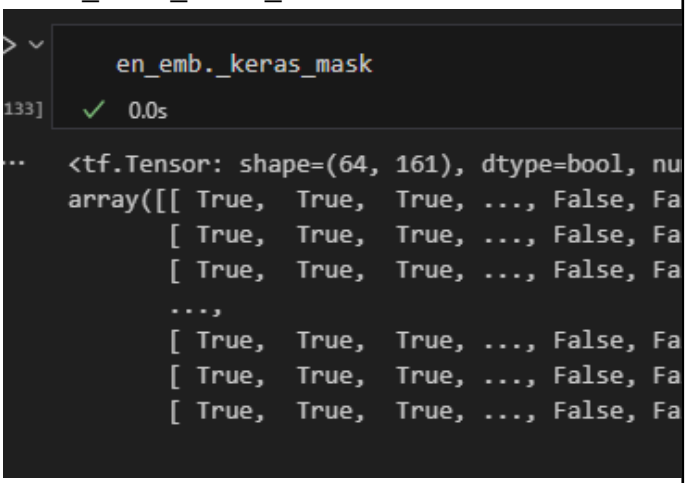
Personal Note

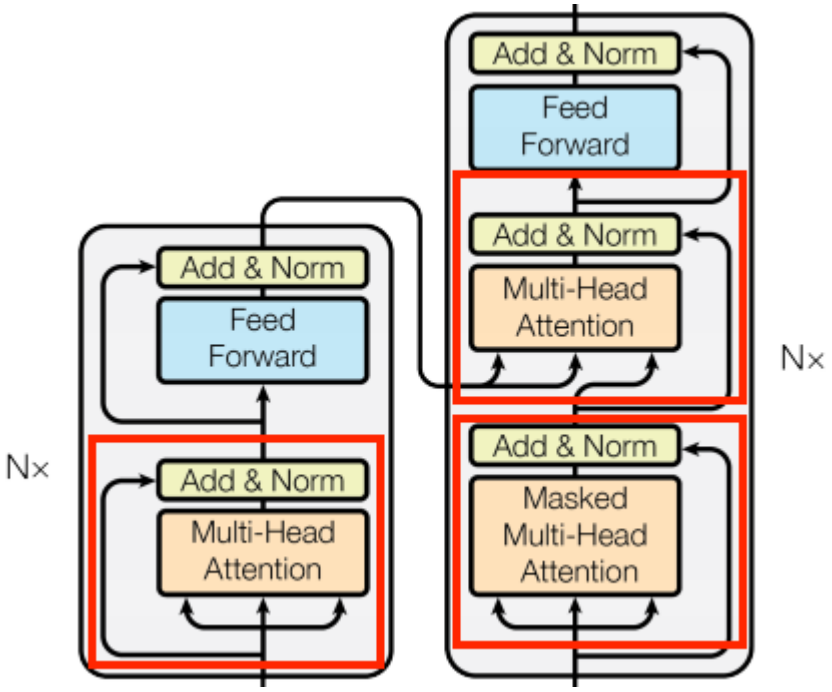
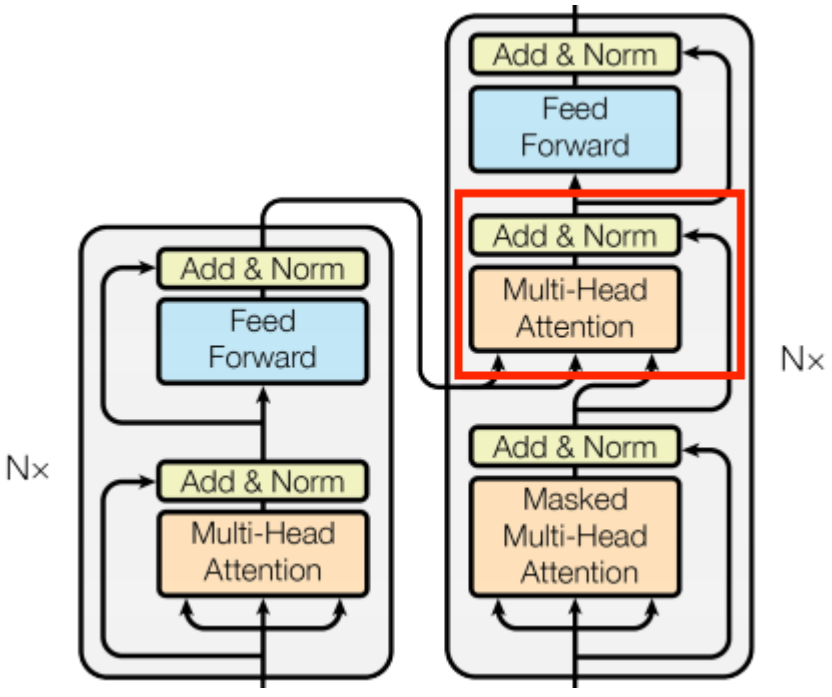
https://www.tensorflow.org/text/tutorials/transformer?authuser=0	Material
https://poloclub.github.io/transformer-explainer/	LLM Transformer Model Visually Explained
https://www.youtube.com/watch?v=wjZofJX0v4M	How large language models work, a visual intro to transformers

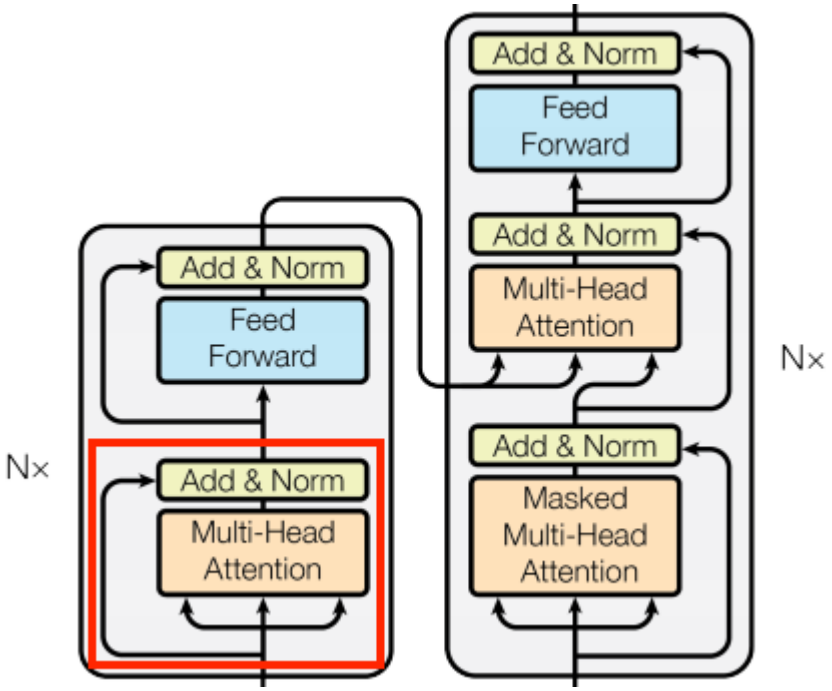
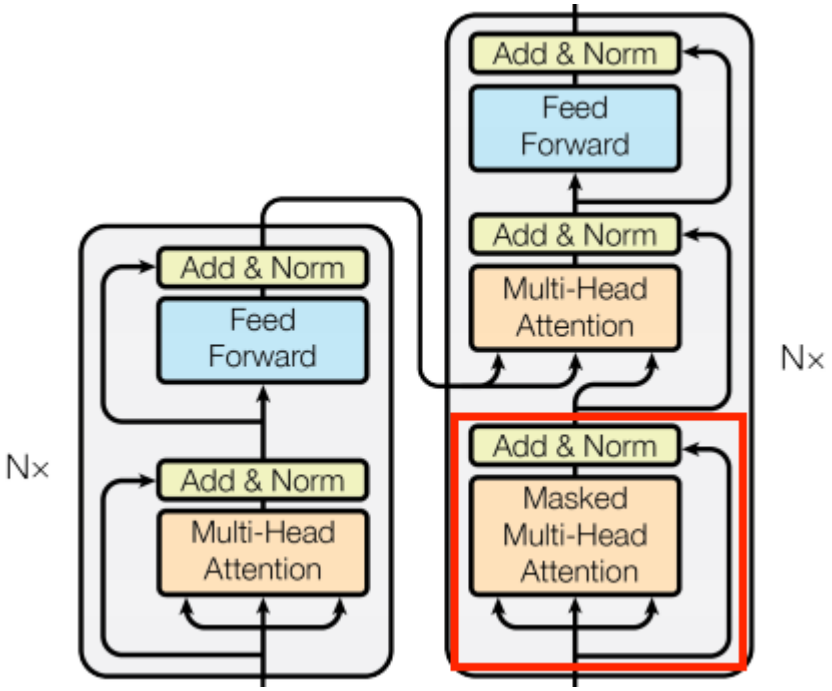
Review lại một số note từ week 6:

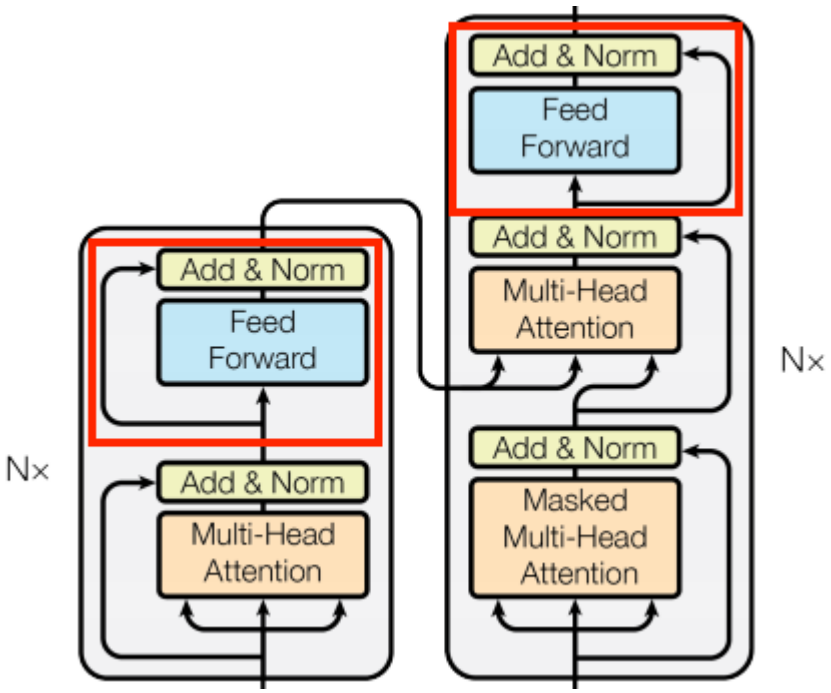
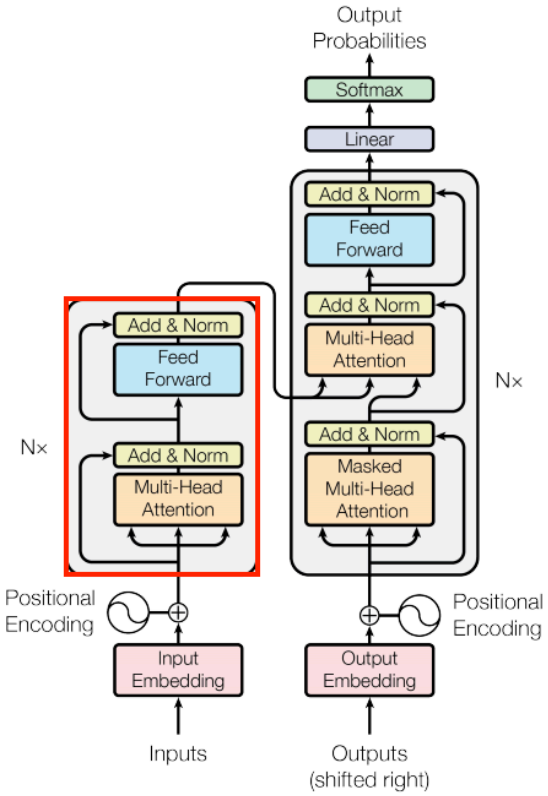
Transformer	Encoder	nhiều lớp, mỗi lớp bao gồm một lớp self-attention và một lớp feedforward (học các features phức tạp từ input)	
	Decoder	tương tự như Encoder nhưng có thêm một lớp Masked Multi-Head Attention để kết nối với output của encoder	
Thành phần chính	Self-Attention	giúp mô hình tập trung vào các từ quan trọng trong câu, cho phép mô hình xác định những từ nào cần "chú ý" trong quá trình dịch.	
	Multi-Head Attention	tập trung vào các phần khác nhau của đầu vào; nắm bắt các mối quan hệ dài hạn	
	Positional Encoding	=> Transformer không có cấu trúc tuần tự như RNN, thêm positional encoding vào để cung cấp tt về vị trí của các token trong chuỗi	
	Feedforward Networks		

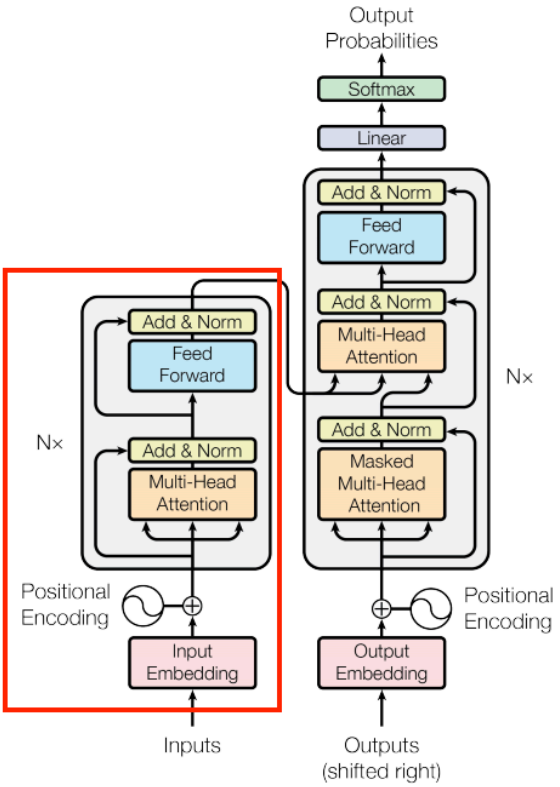
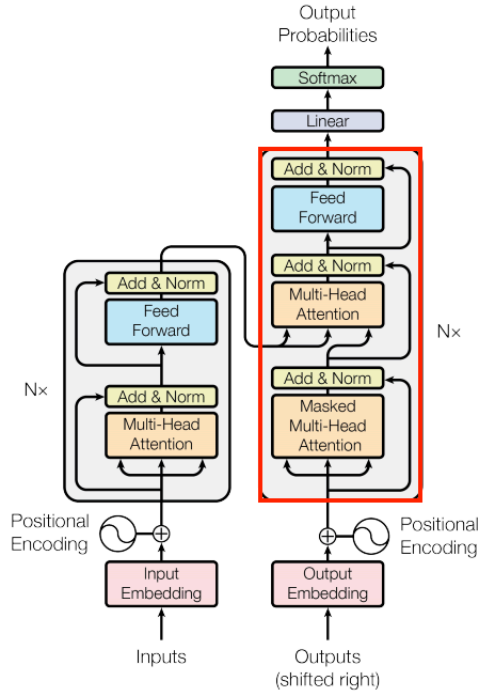
Dataset trong bài Material: TED Talks Open Translation Project, ngôn ngữ Portuguese-English từ tensorflow_datasets (50000 train, 1100 validate và 2000 test)
Các step:

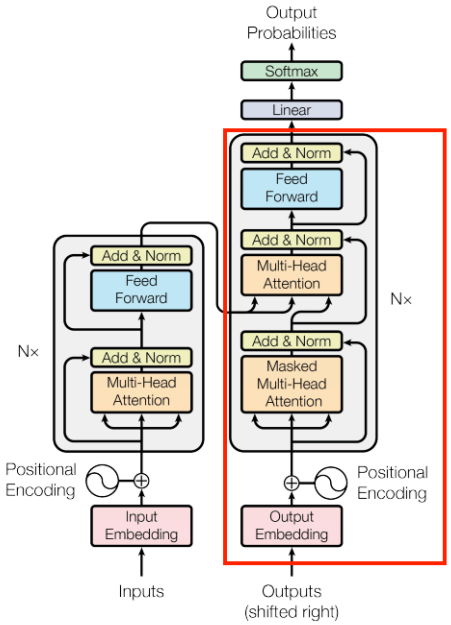
Test the Dataset	Input: (pt,en) Label: chuỗi (en) được dịch chuyển 1 vị trí (=> mỗi token đầu vào tiếng Anh sẽ khớp với nhãn là token tiếp theo) ⇨ "Teacher forcing": bất kể đầu ra của mô hình là gì, chuỗi tiếp theo sẽ luôn dùng giá trị đúng (ground truth) làm đầu vào cho bước tiếp theo ⇨ mô hình huấn luyện nhanh hơn vì không cần xử lý từng bước tuần tự, mà các đầu ra ở các vị trí khác nhau trong chuỗi có thể được tính toán song song	 (trong bài material không có ví dụ, nên em lấy tạm trong code phần assignment)
Define the components	- seq-2-seq - encoder, decoder - self-attention	
Embedding và Positional Encoding	- Dùng lớp tf.keras.layers.Embedding để chuyển đổi từ chuỗi token => vector - Sau khi vector, thêm positional encoding vào vector embedding. Positional encoding dùng hàm sin và cos ở các tần số khác nhau để mã hóa vị trí từ trong chuỗi. ⇨ đảm bảo các vị trí gần nhau có encoding tương đồng, giúp mô hình hiểu thứ tự từ trong câu - en_emb._keras_mask: 	PositionalEmbedding: - vocab_size: kích thước từ vựng - d_model: chiều sâu của vector embedding. - Lớp tf.keras.layers.Embedding dùng để tìm vector embedding của các token. - pos_encoding là giá trị positional encoding được tạo ra trước đó. - Cộng vector embedding và positional encoding lại để có đầu ra hoàn chỉnh. - Dùng en_emb._keras_mask để mô hình bỏ qua các giá trị padding khi huấn luyện. Padding là việc thêm các giá trị đặc biệt (thường là số 0) vào cuối (hoặc đầu) chuỗi để các chuỗi đều có độ dài như nhau. Ví dụ, nếu câu "how are you" được thêm padding để đạt chiều dài 6, ta có thể biểu diễn nó thành "how are you [0] [0] [0]". ⇨ bỏ qua (mask) các vị trí chứa padding khi tính toán, đảm bảo mô hình chỉ học từ các giá trị thật trong chuỗi.

		en_emb._keras_mask là một mask được tạo ra tự động để đánh dấu các vị trí có giá trị padding. Giá trị True trong mask cho biết token thật, còn False cho biết vị trí padding.
Add & Normalize (Residual Connections và Layer Normalization)		
Base Attention Layer		
The cross attention layer		

The global self-attention layer	
The causal self-attention layer	

The feed forward network	
The encoder layer	

The encoder	
The decoder layer	

The decoder		
The transformer		
	Create the Transformer by extending tf.keras.Model	
Hyperparameters	num_layers: số lượng các lớp encoder và decoder trong mô hình Transformer.	Trong bài báo gốc dung 6 num_layers, có thể dung 4 lớp để nhẹ hơn
	d_model: kích thước của các vector embedding	512 trong mô hình gốc, có thể giảm xuống 128
	dff: kích thước của lớp FeedForward trong Transformer.	mô hình gốc là 2048 (trong bài material dung 512)
	num_heads	số lượng "heads" trong lớp attention
	dropout_rate	tỷ lệ dropout để ngăn overfitting
attn_scores	attn_scores = transformer.decoder.dec_layers[-1].last_attn_scores	các trọng số attention từ lớp cuối cùng trong decoder, dùng để theo dõi mức độ mà mô hình tập trung vào từng từ trong câu khi tạo bản dịch
Training		
Set up the optimizer	Use the Adam optimizer with a custom learning rate scheduler - Learning Rate Schedule - Warmup Steps: tăng dần tốc độ học qua một số bước nhất định	Optimizer (Adam) điều chỉnh trọng số của mô hình = cách giảm dần loss dựa trên Learning Rate Schedule tùy chỉnh

	=> mô hình dần quen với quá trình điều chỉnh trọng số trước khi học với tốc độ tối đa, giảm thiểu nguy cơ gradient vanishing/exploding.	Sau một số bước, learning rate tăng dần, sau đó giảm khi mô hình đã qua giai đoạn "khởi động" (warmup).
Set up the loss and metrics	Trong quá trình tính toán loss, cần có mask để loại bỏ các giá trị padding nhằm tránh ảnh hưởng đến kết quả - Masking: loại bỏ các giá trị padding trong dữ liệu đầu vào. (Trong neural machine translation, các câu đầu vào có độ dài khác nhau => padding => khi tính loss và accuracy, cần bỏ qua giá trị này => tập trung vào những từ thực tế trong câu) - Sparse Categorical Crossentropy: hàm mất mát thích hợp cho các bài toán phân loại nhiều lớp với nhãn đầu ra được mã hóa dưới dạng số nguyên - Attention Weights: mức độ liên quan giữa từ ở ngôn ngữ nguồn và từ đang được dịch	masked_loss & masked_accuracy
Train the model	transformer.compile	
	transformer.fit	