The custom_vit_decoder.py file defines the ImageCaptioningModel, which consists of a Vision Transformer (ViT) encoder (ViTEncoder) and a transformer decoder (TransformerDecoder). These components rely on hyperparameters defined in a config module and additional parameters defined within the classes. Below, I'll list all the parameters and hyperparameters for both the ViT encoder and the transformer decoder, explain their roles, and clarify whether they are model parameters (learnable weights) or hyperparameters (fixed settings). The explanations will be concise yet comprehensive, tailored to their use in the image captioning system for the COCO dataset.

ViT Encoder Parameters and Hyperparameters

The ViT encoder is implemented in the ViTEncoder class, which includes a PatchEmbedding module and a series of TransformerEncoderLayers. Below are the parameters and hyperparameters, organized by component.

1. PatchEmbedding (Part of ViTEncoder)

Hyperparameters (from config or class initialization):

- **img_size** (default: config.IMAGE_SIZE[0], e.g., 224)
  - **Explanation**: The height/width of input images (assumed square). For COCO images, this is typically 224x224 pixels, determining the input resolution.
  - **Role**: Defines the image size for patch extraction, affecting the number of patches.
- **patch_size** (default: config.PATCH_SIZE, e.g., 16)
  - **Explanation**: The size of each square patch (e.g., 16x16 pixels). Images are divided into non-overlapping patches.
  - **Role**: Controls the granularity of image division, impacting the number of patches and model complexity.
- **in_channels** (default: 3)
  - **Explanation**: Number of input channels (3 for RGB images, as used in COCO).
  - **Role**: Specifies the input depth for the convolutional layer that projects patches to embeddings.
- **embed_dim** (default: config.EMBED_DIM, e.g., 768)
  - **Explanation**: The dimensionality of patch and [CLS] token embeddings.
  - **Role**: Determines the size of the feature vectors processed by the transformer, affecting model capacity.
- **n_patches** (derived: (img_size // patch_size) ** 2, e.g., 196 for 224x224 images with 16x16 patches)
  - **Explanation**: The number of patches extracted from the image, calculated as the square of the image size divided by patch size.
  - **Role**: Defines the sequence length (plus [CLS] token) for the transformer encoder (e.g., 196 patches + 1 [CLS] = 197).

Parameters (learnable weights):

- **projection** (nn.Conv2d(in_channels, embed_dim, kernel_size=patch_size, stride=patch_size))
    - **Explanation**: A convolutional layer that projects each patch to an embed_dim-dimensional vector.
    - **Shape**: Weights: [embed_dim, in_channels, patch_size, patch_size] (e.g., [768, 3, 16, 16]), Bias: [embed_dim] (e.g., [768]).
    - **Role**: Transforms raw pixel patches into embedded representations.
- **cls_token** (nn.Parameter(torch.randn(1, 1, embed_dim)))
    - **Explanation**: A learnable [CLS] token prepended to the patch sequence, used to aggregate global image information (similar to BERT's [CLS] token).
    - **Shape**: [1, 1, embed_dim] (e.g., [1, 1, 768]).
    - **Role**: Captures a global image representation for the decoder.
- **pos_embed** (nn.Parameter(torch.randn(1, n_patches + 1, embed_dim)))
    - **Explanation**: Learnable positional embeddings for each patch and the [CLS] token.
    - **Shape**: [1, n_patches + 1, embed_dim] (e.g., [1, 197, 768]).
    - **Role**: Adds positional information to patches, enabling the transformer to distinguish their spatial arrangement.

2. TransformerEncoderLayer

Hyperparameters:

- **embed_dim** (default: config.EMBED_DIM, e.g., 768)
    - **Explanation**: The dimensionality of input/output embeddings, matching PatchEmbedding.
    - **Role**: Defines the size of feature vectors processed in attention and feed-forward layers.
- **num_heads** (default: 12)
    - **Explanation**: Number of attention heads in multi-head attention.
    - **Role**: Controls parallelism in attention, with embed_dim / num_heads (e.g., 768 / 12 = 64) as the head dimension.
- **ff_dim** (default: 3072)
    - **Explanation**: The hidden dimension of the feed-forward network (FFN) in each encoder layer.
    - **Role**: Determines the capacity of the FFN, typically 4x embed_dim for transformers.
- **dropout** (default: 0.1)
    - **Explanation**: Dropout rate applied after attention and FFN layers.
    - **Role**: Prevents overfitting by randomly dropping activations during training.

Parameters:

- **attn.qkv** (nn.Linear(embed_dim, embed_dim * 3))
    - **Explanation**: Linear layer generating query, key, and value vectors for multi-head attention.
    - **Shape**: Weights: [embed_dim * 3, embed_dim] (e.g., [2304, 768]), Bias: [embed_dim * 3] (e.g., [2304]).
    - **Role**: Projects input embeddings into Q, K, V for attention computation.

- **attn.proj** (nn.Linear(embed_dim, embed_dim))
    - **Explanation**: Linear layer projecting attention output back to embed_dim.
    - **Shape**: Weights: [embed_dim, embed_dim] (e.g., [768, 768]), Bias: [embed_dim] (e.g., [768]).
    - **Role**: Combines multi-head attention outputs.
- **ff** (nn.Sequential(nn.Linear(embed_dim, ff_dim), nn.GELU(), nn.Linear(ff_dim, embed_dim)))
    - **Explanation**: Feed-forward network with two linear layers and GELU activation.
    - **Shape**:
        - First nn.Linear: Weights: [ff_dim, embed_dim] (e.g., [3072, 768]), Bias: [ff_dim] (e.g., [3072]).
        - Second nn.Linear: Weights: [embed_dim, ff_dim] (e.g., [768, 3072]), Bias: [embed_dim] (e.g., [768]).
    - **Role**: Applies position-wise transformations to enhance feature representations.
- **norm1, norm2** (nn.LayerNorm(embed_dim))
    - **Explanation**: Layer normalization applied after attention and FFN.
    - **Shape**: Weights and Bias: [embed_dim] (e.g., [768]) for each.
    - **Role**: Stabilizes training by normalizing activations.

3. ViTEncoder

Hyperparameters:

- **num_layers** (default: 12)
    - **Explanation**: Number of transformer encoder layers.
    - **Role**: Controls the depth of the encoder, affecting its capacity to model complex image features.
- Inherits img_size, patch_size, in_channels, embed_dim, num_heads, ff_dim from PatchEmbedding and TransformerEncoderLayer.

Parameters:

- **patch_embed** (parameters from PatchEmbedding): projection, cls_token, pos_embed.
- **layers** (parameters from num_layers instances of TransformerEncoderLayer): attn.qkv, attn.proj, ff, norm1, norm2 for each layer.
- **norm** (nn.LayerNorm(embed_dim))
    - **Explanation**: Final layer normalization after all encoder layers.
    - **Shape**: Weights and Bias: [embed_dim] (e.g., [768]).
    - **Role**: Normalizes the final encoder output.

Transformer Decoder Parameters and Hyperparameters

The transformer decoder is implemented in the TransformerDecoder class, with TransformerDecoderLayers for processing tokenized captions.

1. TransformerDecoderLayer

Hyperparameters:

- **embed_dim** (default: config.DEC_EMBED_DIM, e.g., 512)
    - **Explanation**: Dimensionality of decoder embeddings, potentially different from encoder's embed_dim.
    - **Role**: Defines the size of token embeddings and attention outputs.
- **num_heads** (default: 8)
    - **Explanation**: Number of attention heads in self-attention and cross-attention.
    - **Role**: Controls parallelism, with embed_dim / num_heads (e.g., 512 / 8 = 64) as the head dimension.
- **ff_dim** (default: 2048)
    - **Explanation**: Hidden dimension of the feed-forward network.
    - **Role**: Determines FFN capacity, typically 4x embed_dim.
- **dropout** (default: 0.1)
    - **Explanation**: Dropout rate for attention and FFN layers.
    - **Role**: Prevents overfitting.

Parameters:

- **self_attn.qkv, cross_attn.qkv** (nn.Linear(embed_dim, embed_dim * 3))
    - **Explanation**: Linear layers for query, key, value in self-attention and cross-attention.
    - **Shape**: Weights: [embed_dim * 3, embed_dim] (e.g., [1536, 512]), Bias: [embed_dim * 3] (e.g., [1536]) for each.
    - **Role**: Generates Q, K, V for self-attention (within captions) and cross-attention (with encoder memory).
- **self_attn.proj, cross_attn.proj** (nn.Linear(embed_dim, embed_dim))
    - **Explanation**: Projection layers for attention outputs.
    - **Shape**: Weights: [embed_dim, embed_dim] (e.g., [512, 512]), Bias: [embed_dim] (e.g., [512]) for each.
    - **Role**: Combines attention outputs.
- **ff** (nn.Sequential(nn.Linear(embed_dim, ff_dim), nn.GELU(), nn.Linear(ff_dim, embed_dim)))
    - **Explanation**: Feed-forward network.
    - **Shape**:
        - First nn.Linear: Weights: [ff_dim, embed_dim] (e.g., [2048, 512]), Bias: [ff_dim] (e.g., [2048]).
        - Second nn.Linear: Weights: [embed_dim, ff_dim] (e.g., [512, 2048]), Bias: [embed_dim] (e.g., [512]).
    - **Role**: Enhances token representations.
- **norm1, norm2, norm3** (nn.LayerNorm(embed_dim))
    - **Explanation**: Layer normalization after self-attention, cross-attention, and FFN.
    - **Shape**: Weights and Bias: [embed_dim] (e.g., [512]) for each.
    - **Role**: Stabilizes training.

2. TransformerDecoder

Hyperparameters:

- **vocab_size** (default: config.VOCAB_SIZE, e.g., 30,000)
    - **Explanation**: Size of the vocabulary, matching the SentencePiece tokenizer's vocabulary (from tokenizer.vocab).
    - **Role**: Determines the output dimension for token predictions.
- **embed_dim** (default: config.DEC_EMBED_DIM, e.g., 512)
    - **Explanation**: Dimensionality of token embeddings.
    - **Role**: Defines the size of decoder inputs/outputs.
- **num_layers** (default: 6)
    - **Explanation**: Number of decoder layers.
    - **Role**: Controls decoder depth, affecting caption generation capacity.
- **num_heads** (default: 8)
    - **Explanation**: Number of attention heads per layer.
    - **Role**: Same as in TransformerDecoderLayer.
- **ff_dim** (default: 2048)
    - **Explanation**: Feed-forward network hidden dimension.
    - **Role**: Same as in TransformerDecoderLayer.
- **max_seq_len** (default: config.MAX_SEQ_LEN, e.g., 50)
    - **Explanation**: Maximum length of generated captions.
    - **Role**: Limits positional embeddings and sequence length during generation.

Parameters:

- **token_embed** (nn.Embedding(vocab_size, embed_dim))
    - **Explanation**: Embeds token IDs into embed_dim-dimensional vectors.
    - **Shape**: Weights: [vocab_size, embed_dim] (e.g., [30,000, 512]).
    - **Role**: Converts input token IDs (from tokenizer) to embeddings.
- **pos_embed** (nn.Parameter(torch.randn(1, max_seq_len, embed_dim)))
    - **Explanation**: Learnable positional embeddings for caption tokens.
    - **Shape**: [1, max_seq_len, embed_dim] (e.g., [1, 50, 512]).
    - **Role**: Adds positional information to token embeddings.
- **layers** (parameters from num_layers instances of TransformerDecoderLayer): self_attn, cross_attn, ff, norm1, norm2, norm3 for each layer.
- **norm** (nn.LayerNorm(embed_dim))
    - **Explanation**: Final layer normalization.
    - **Shape**: Weights and Bias: [embed_dim] (e.g., [512]).
    - **Role**: Normalizes decoder output.
- **proj** (nn.Linear(embed_dim, vocab_size))
    - **Explanation**: Projects decoder output to vocabulary logits.
    - **Shape**: Weights: [vocab_size, embed_dim] (e.g., [30,000, 512]), Bias: [vocab_size] (e.g., [30,000]).
    - **Role**: Produces probabilities for next-token prediction.

ImageCaptioningModel Parameters and Hyperparameters

The ImageCaptioningModel ties the ViT encoder and transformer decoder together, with an additional projection layer.

Hyperparameters:

- **vocab_size** (default: config.VOCAB_SIZE, e.g., 30,000)
    - **Explanation**: Vocabulary size for the decoder, matching the tokenizer.
    - **Role**: Defines the output space for caption generation.
- **img_size** (default: config.IMAGE_SIZE[0], e.g., 224)
    - **Explanation**: Image size for the encoder.
    - **Role**: Same as in PatchEmbedding.
- **patch_size** (default: config.PATCH_SIZE, e.g., 16)
    - **Explanation**: Patch size for the encoder.
    - **Role**: Same as in PatchEmbedding.
- **embed_dim** (default: config.EMBED_DIM, e.g., 768)
    - **Explanation**: Encoder embedding dimension.
    - **Role**: Same as in ViTEncoder.
- **dec_embed_dim** (default: config.DEC_EMBED_DIM, e.g., 512)
    - **Explanation**: Decoder embedding dimension, potentially different from embed_dim.
    - **Role**: Allows flexibility in decoder capacity, aligned with projection layer.

Parameters:

- **encoder** (parameters from ViTEncoder): Includes all parameters from PatchEmbedding and TransformerEncoderLayers.
- **projection** (nn.Linear(embed_dim, dec_embed_dim))
    - **Explanation**: Projects encoder output to decoder's embedding dimension.
    - **Shape**: Weights: [dec_embed_dim, embed_dim] (e.g., [512, 768]), Bias: [dec_embed_dim] (e.g., [512]).
    - **Role**: Aligns encoder and decoder feature dimensions.
- **decoder** (parameters from TransformerDecoder): Includes all parameters from TransformerDecoderLayers, token_embed, pos_embed, norm, proj.

Integration with COCO and Tokenizer

- **COCO Dataset**:
    - img_size (e.g., 224) and in_channels (3) are tailored for COCO's RGB images.
    - vocab_size (e.g., 30,000) matches the SentencePiece tokenizer trained on COCO captions (tokenizer.vocab).
    - max_seq_len (e.g., 50) is set to accommodate typical COCO caption lengths.
- **Tokenizer**:

- The decoder’s token_embed and proj layers use vocab_size from the tokenizer, ensuring compatibility with tokens like <BOS>, <EOS>, <PAD> (IDs 3, 4, 5).
- The generate method in ImageCaptioningModel uses the tokenizer’s bos_id and eos_id for autoregressive caption generation.

Summary Table

Component	Hyperparameters	Parameters	Explanation
PatchEmbedding	img_size, patch_size, in_channels, embed_dim, n_patches (derived)	projection, cls_token, pos_embed	Converts images to patch embeddings with positional information.
TransformerEncoderLayer	embed_dim, num_heads, ff_dim, dropout	attn.qkv, attn.proj, ff, norm1, norm2	Processes patch embeddings with self-attention and FFN.
ViTEncoder	num_layers, plus inherited from above	patch_embed, layers, norm	Encodes images into contextualized embeddings.
TransformerDecoderLayer	embed_dim, num_heads, ff_dim, dropout	self_attn.qkv, self_attn.proj, cross_attn.qkv, cross_attn.proj, ff, norm1, norm2, norm3	Processes captions with self-attention and cross-attention to encoder output.
TransformerDecoder	vocab_size, embed_dim, num_layers, num_heads, ff_dim, max_seq_len	token_embed, pos_embed, layers, norm, proj	Generates caption token probabilities.
ImageCaptioningModel	vocab_size, img_size, patch_size, embed_dim, dec_embed_dim	encoder, projection, decoder	Combines encoder and decoder for end-to-end image captioning.

Example Values and Impact

- **ViT Encoder** (e.g., img_size=224, patch_size=16, embed_dim=768, num_layers=12, num_heads=12, ff_dim=3072):
    - Processes 224x224 COCO images into 196 patches + 1 [CLS] token, producing [batch_size, 197, 768] embeddings.
    - High num_layers and num_heads ensure robust feature extraction, suitable for complex images.
- **Transformer Decoder** (e.g., vocab_size=30,000, embed_dim=512, num_layers=6, num_heads=8, ff_dim=2048, max_seq_len=50):
    - Handles COCO captions with a large vocabulary and moderate sequence length.
    - Fewer layers/heads than the encoder reduce computational cost, as decoding is autoregressive.
- **Projection Layer**: Bridges embed_dim (768) to dec_embed_dim (512), allowing flexibility in model design.

Notes

- **Config Dependency**: Most hyperparameters are sourced from the config module (e.g., config.IMAGE_SIZE, config.VOCAB_SIZE), centralizing settings for consistency with train_custom_model.py and tokenizer.py.
- **Learnable Parameters**: The model has millions of parameters (e.g., ViT with 12 layers and decoder with 6 layers), typical for transformer-based models, trained from scratch on COCO (no pre-trained weights indicated).
- **COCO Specificity**: Hyperparameters like img_size, vocab_size, and max_seq_len are tailored for COCO's image-caption pairs, aligning with the tokenizer's vocabulary.

If you need a parameter count, a sample configuration with exact values, or a deeper dive into any component's role, let me know!