1. **Model Architecture:**

- I**mage Processing Component**: Specify the architecture of the CNN model to be used for image feature extraction. Detail the type and number of layers, activation functions, and any regularization techniques.

**CNN Model:**

* Input Layer 224x224x3: Size chosen due to standardization and its use in pre-trained models like VGG-16.

Two convolutional layers are proposed as a starting point. This is a middle point between using a single layer for extraction and tree or more layers. As the number of layers grows, the number of features to be extracted increases along with the computational cost.

* First Convolutional Layer:
  + 32 Filters, Kernel Size (3,3), ReLu Activation Function, Padding of 1, Stride of 1.
  + With these parameters, the image maintains its original size.
  + Output of (224,224,32)
* First Batch Normalization Layer: Normalization of the outputs from the convolutional layers. The normalization of the data helps to reduce the training time of the model
  + Output of (224,224,32)
* Second convolutional layer:
  + 64 filters, Kernel Size (3,3), ReLu Activation function, Padding of 1, Stride of 1.
  + Output of (224,224,64)
* Second Batch Normalization Layer: Normalization of the outputs from the convolutional layers. The normalization of the data helps to reduce the training time of the model
  + Output of (224,224,32)
* Pooling Layer: Reduced the computational load by reducing the matrix sizes by half.
  + Max Pooling, Pool size of (2,2), Stride of 2
  + Output of (112,112,64)
* Dropout Layer: Prevent overfitting by randomly deactivating neurons, and force robust learning from features.
  + Output (112,112,64)
* Fully Connected Layer: Ensures the integration of the learned features from previous layers.
  + ReLu function used on each neuron
  + Output (256,1), assuming we have a complex task to caption the input images. However to ensure which is the best fit for the problem, we would need to run tests and observe the metrics to update this to 128 if the task is simpler. If the task turns out to require more neurons in this layer we might have to assess the computational resources available before updating this design parameter.

The output of (256,1) should have all the features learned from the images and is the input for the RNN that will process the captions.

- **Text Generation Component**: Outline the RNN/LSTM architecture for generating captions. Specify the architecture details including the number of layers, cell type (RNN, LSTM, GRU), and any other relevant parameters.

- **Integration**: Explain how you will integrate the CNN and RNN components for the task of image captioning.

Aquí hay que explicar que el output de la fully connected network es la entrada para una RNN que hará el captioning de las imágenes. Esto lo describo en el último párrafo de la sección anterior. Dejo descripción de la parte que sigue para que no ea tan complicado definir los detalles restantes. el primer módulo creo yo tiene suficiente detalle para ser definitivo.

Lo siguiente además de definir las siguientes partes a detalle es hacer diagramas que ilustren lo que estamos proponiendo.

For the definition of the RNN model we need to also define the vocabulary of the model. Let’s assume 5000(?) tokens.

**RNN model:**

Input: (256,1) feature vector from the CNN model.

Embedding Layer: Converts word indices to dense vectors (let's say 300 dimensions).

LSTM Layers: One or two LSTM layers with 256 units.

Output Layer: A dense layer with a softmax activation, outputting probabilities over the vocabulary.

**Integration for Image captioning:**

Data Preparation: Your dataset should consist of images and their corresponding captions. The captions need to be preprocessed (tokenized, padded or truncated to a fixed length, and converted to sequences of word indices).

Feature Extraction: Pass the images through the CNN to get the feature vectors.

Caption Generation: Use the RNN to generate captions. For training, you will use teacher forcing (feeding the correct word as the next input). For prediction, you will use the RNN's output at each time step as the next input.

**Training the model:**

Loss Function: Categorical cross-entropy.

Optimizer: A standard optimizer like Adam.

Training: During training, you'll feed both the image features and the caption sequences into the RNN.

**Generation of captions:**

Start Token: Begin with a start-of-sequence token.

Iteration: At each step, use the RNN to predict the next word until an end-of-sequence token is generated or a maximum length is reached.

2. **Training Specifications**:

**- Data Preprocessing**: Describe how you would compile the dataset (use a public dataset or generate your own, number of samples, etc.). Also describe how you will preprocess the image and text data.

**- Training Regimen**: Suggest the number of training epochs, batch size, and any data augmentation strategies.

**- Optimizer and Learning Rate**: Choose an appropriate optimizer and learning rate strategy, justifying your choices.

**3. Performance Metrics:**

- Specify which metrics you would use to evaluate the performance of your image captioning model.

**4. Additional Considerations:** - Discuss any potential challenges and how you might address them.

Referencias:

* <https://keras.io/api/applications/vgg/>
* Geron, A. (2019). Hands-on machine learning with scikit-learn, keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems (2nd ed.). O’Reilly Media.