* **Converting integer representations to one hot encoding**: We had 5 different categorical features representing each nucleotide. In machine learning, several models tend to perform better with numerical features. One hot encoding provides a way to convert these features into numerical features represented by vectors. When dealing with categorical variables represented as integers, the model may assume ordinal relationships between the integers. One-hot encoding eliminates this issue by representing categories as independent dimensions, preventing the model from inferring ordinal relationships.
* **Model Architecture**:
  + **Initial CNN Layers**: CNNs are powerful at capturing local patterns and features in data through convolutional and pooling operations. They excel at extracting hierarchical representations of input data. CNN layers are effective in capturing local context and dependencies. This can be beneficial for learning short-term dependencies in sequential data. In simpler words, CNN layers act as a feature extraction layer that extracts the features from the data, which can then be fed to the LSTM layers.
  + **ReLU Layer:** ReLU introduces non-linearity to the model. Convolutional layers, by themselves, are linear operations (convolutions and matrix multiplications). Introducing non-linear activation functions, such as ReLU, allows the model to learn and represent more complex and expressive mappings between inputs and outputs.
  + **Maxpool Layer:** Max pooling is a downsampling operation that reduces the spatial dimensions of the input tensor. It retains the maximum value in each local region, providing translation invariance and reducing the computational load. Max pooling with a kernel\_size of 1 does not perform any pooling operation since it considers only one element at a time, effectively maintaining the original values. Although, it is still kept in the architecture and can be used with a bigger kernel size when working with a large dataset.
  + **LSTM Layer**: LSTM layers are well-suited for capturing longer-term dependencies. Combining CNNs with LSTMs allows the model to capture both local and global context in the data.LSTM stands for long short-term memory, and it allows the model to capture the long-term dependencies present in the data. A bidirectional LSTM focuses on both forward and backward data. The key characteristic of bidirectional LSTMs is their ability to capture information from the past (preceding time steps) as well as the future (subsequent time steps) for each point in the input sequence. This bidirectional processing enhances the network's ability to capture contextual information from both directions.
  + **Final Linear Layer:** Since we have to make a single prediction for the entire data, our model can be thought of as a regression task. Therefore, a final linear layer with a single output is the best option for the model architecture.
  + **Dropout Layer:** Dropout is a regularization technique used to prevent overfitting in neural networks. During training, randomly selected neurons are "dropped out" (i.e., their outputs are set to zero) on each forward pass. This helps prevent the network from relying too much on specific neurons and encourages a more robust and generalized learning.
* **Hyper Parameters:**
  + **Batch Size:** A standardized batch size of 32 is used for the model. The batch size of 32 seemed to work well with the data. Smaller batch sizes can potentially lead to better generalization as the model updates its parameters more frequently, allowing it to adapt to different patterns in the data. Larger batch sizes can take advantage of the parallel processing capabilities of modern GPUs, leading to faster training times. Larger batch sizes provide smoother gradient estimates, which can lead to more stable convergence.
  + **CNN\_Channels:** Increasing the number of channels increases the model's capacity to learn complex features from the input data. However, a very high number of channels might lead to overfitting, especially if the dataset is not large enough. Training a model with a higher number of channels requires more computational resources, including GPU memory. Given the small dataset, choosing 128 as the number of channels worked well with the model. The given number of channels was successfully able to capture the intricacies of the data without leading to overfitting.
  + **Kernel Size:** A smaller kernel size, such as 2, is effective for capturing local features and patterns in the input data. It allows the convolutional layer to focus on adjacent pairs of values in the input sequence. Kernel size 2 means a 1-dimensional kernel with a width of 2. It is a sequence of weights applied to adjacent pairs of elements in the input sequence. The convolution operation slides this 1D kernel across the input sequence, capturing local patterns or features. A kernel size of 2 makes the convolutional layer sensitive to pairs of adjacent values in the input sequence. This can be suitable for tasks where the model needs to capture dependencies between closely located elements.
  + **Hidden Size:** The hidden size influences the capacity of the LSTM to capture and store information from the input sequence. A larger hidden size allows the model to capture more complex patterns, but it also increases the number of parameters in the model. The complexity of the task at hand can impact the choice of the hidden size. For more complex tasks or datasets with intricate patterns, a larger hidden size might be beneficial. For our data, the ‘128’ hidden size worked pretty well, and therefore using a higher number was avoided to keep the computational requirement low.
  + **Number of Stacked Layers:** The number of stacked layers determines the depth of the LSTM network. Stacking multiple LSTM layers allows the model to learn hierarchical representations of the input sequence. Increasing the number of stacked layers generally enhances the model's ability to capture long-term dependencies in sequential data. Each layer can capture different levels of abstraction and temporal patterns. A higher number of stacked layers increases the overall complexity of the model. This can be beneficial for tasks that require a deeper understanding of the input sequence. For our dataset, 3 stacked layers were sufficient for the model to converge effectively.
  + **Learning Rate:** A smaller learning rate, such as 0.0001, means that the model parameters are updated with smaller steps during each iteration. This can result in more stable convergence but might require more iterations to reach the optimal solution. The learning rate influences the speed at which the model converges during training. Smaller learning rates may lead to slower convergence, but they can be beneficial for achieving a more precise solution. To keep the model training time controlled, a learning rate of 0.0002 was used.
  + **Number of Epochs:** The number of epochs is a hyperparameter that defines the total number of times a machine learning model will iterate over the entire training dataset during training. In the provided code, the number of epochs is not explicitly specified. The training loop is structured to iterate over batches of the training data until the total number of iterations is achieved. Our model seemed to converge after 30 iterations. For safety, a value of 50 was chosen.
* **Model optimizer:** The torch.optim.Adam optimizer is a popular optimization algorithm that combines ideas from both momentum optimization and RMSprop. It is well-suited for training deep neural networks and is known for its efficiency and robustness. We continued with the standard optimizer for sequential data. Weight\_decay=1e-4 is an optional parameter that introduces L2 regularization (weight decay) during optimization. It penalizes large weights to prevent overfitting.
* **MSE Loss Function:** MSE is well-suited for regression tasks where the model aims to predict a continuous numeric value. In the context of DNA sequencing, where the final layer of the model is a linear layer producing a single output (the number of CpGs pairs), MSE is a natural choice.