Ayesha Ashraf

Task 27

Stacking up layer in neural network

Recurrent Stacking of Layers in Neural Networks:

Stacked recurrent layers are defined as the Recurrent Neural Network model that is comprised of multiple Recurrent Neural Network layers. The Recurrent neural network layer provides a sequence to the output rather than a single value output to Recurrent neural network layer. Stacking recurrent layers are a way of building more powerful recurrent neural networks. Stacking recurrent layers are used to increase the capacity of a neural network until overfitting becomes a primary obstacle that is even after using dropout to mitigate the overfitting. Stacking recurrent layers increases the representational power of the neural network at a cost of higher computational loads.

This recipe explains what are staking recurrent layers, how it is beneficial for neural network models and how it can be executed.

Explanation of Stacking recurrent layers.

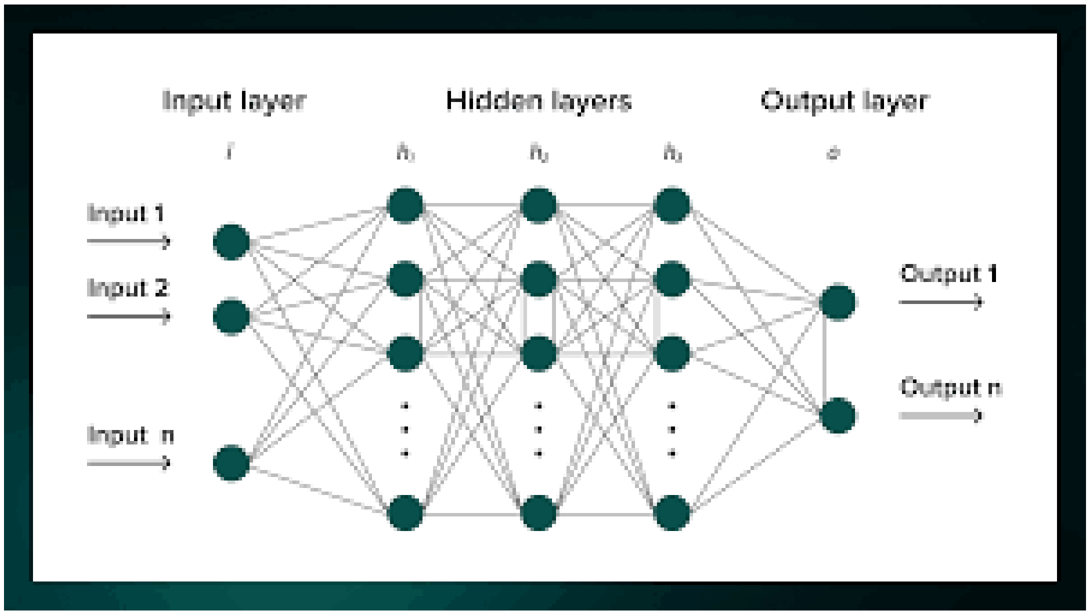
Stacking recurrent layers on the top of each other in Keras, all the intermediate layers should return their full sequence of the outputs that is a 3D tensor, rather than their output at last timestep. This is done by specifying return_sequences = TRUE command.

Stacking recurrent layers can be executed using the Keras framework easily. It increases the complexity of the neural network model. It decreases the validation loss and helps in improving model accuracy.

In stacking, an algorithm takes the outputs of sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction.

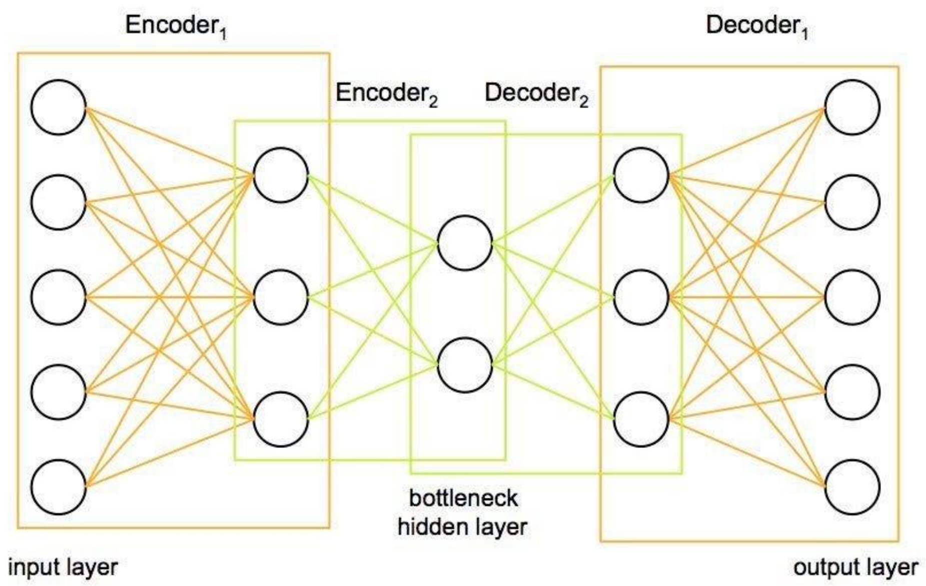
It may be helpful to think of the stacking procedure as having two levels: level 0 and level 1.

- **Level 0**: The level 0 data is the training dataset inputs and level 0 models learn to make predictions from this data.
- **Level 1**: The level 1 data takes the output of the level 0 models as input and the single level 1 model, or meta-learner, learns to make predictions from this data.



An Application to Neural Machine Translation

In deep neural network modeling, the most common practice is to stack a number of recurrent, convolutional, or feed-forward layers in order to obtain high-quality continuous space representations which in turn improves the quality of the network's prediction. Conventionally, each layer in the stack has its own parameters which leads to a significant increase in the number of model parameters. In this paper, we propose to share parameters across all layers thereby leading to a recurrently stacked neural network model. We report on an extensive case study on neural machine translation (NMT), where we apply our proposed method to an encoder-decoder based neural network model, i.e., the Transformer model, and experiment with three Japanese--English translation datasets. We empirically demonstrate that the translation quality of a model that recurrently stacks a single layer 6 times, despite having significantly fewer parameters, approaches that of a model that stacks 6 layers where each layer has different parameters. We also explore the limits of recurrent stacking where we train extremely deep NMT models. This paper also examines the utility of our recurrently stacked model as a student model through transfer learning via leveraging pre-trained parameters and knowledge distillation, and shows that it compensates for the performance drops in translation quality that the direct training of recurrently stacked model brings. We also show how transfer learning helps in faster decoding on top of the already reduced number of parameters due to recurrent stacking. Finally, we analyze the effects of recurrently stacked layers by visualizing the attentions of models that use recurrently stacked layers and models that do not.



Encoder$_1$
Decoder$_1$
Encoder$_2$
Decoder$_2$
bottleneck hidden layer
input layer
output layer