301-Project

Converting Hand-Written Equation to LaTeX Expression

CNN with Positional Decoder and Attention Bi-LSTM Encoder Architecture

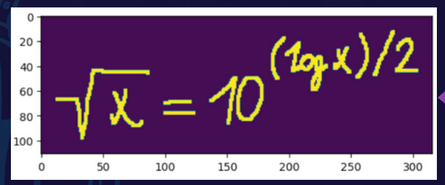
Group Member: Zach Wu, Kelly Deng, Daniel Xu

Date: 2024/05/07 10:36AM

Github Repo: https://github.com/zachwu123/image2latex

Executive Summary

Input: Hand-writting Math Symbol
Output: LaTex Format Equation




```
u ( \theta _ { 0 } , p ) = 0
a _ { i , j } = a _ { i - 1 , j + 1 }
d ^ { 3 } u
```

Technical Challenge:
Adapt to the CNN decoder and RNN encoder structure

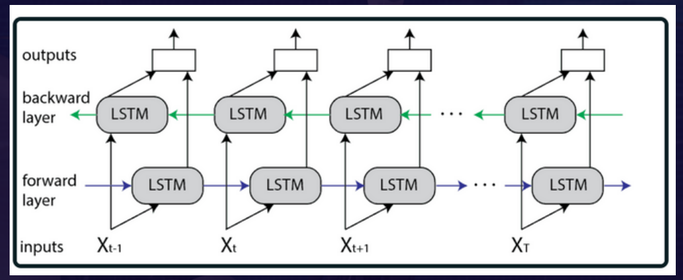
Approaches:

- ResNet50 + LSTM
- ResNet50 + GRU
- ResNet50 + GRU
- DenseNet + LSTM
- DenseNet + GRU
- DenseNet + positional

  ……

Final Solution:

- CNN + positional + Bi-LSTM with Attention
- No dense / skip connection in the decoder



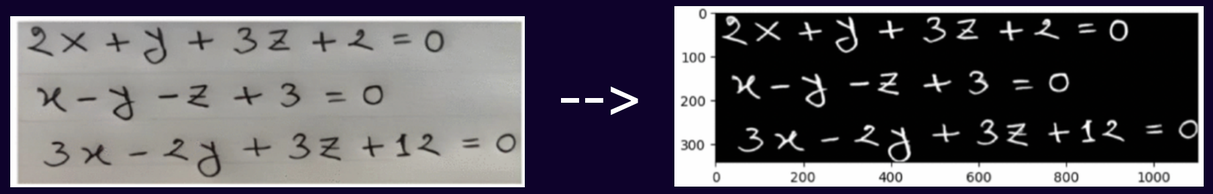
Benefit Of Solution:

- Sequence Order Preservation (Positional)
- Contextual Understanding (Bi-LSTM)
- Focused Accuracy (Attention- additive)
- Better Performance Given Limited Data

Related Works

Shivangi, R. K. Sah, Shreeyam, G. Florance and M. Nirmala, "CNNCalc – An Implementation of a Handwritten Mathematical Equation Solver"

- Image Preprocessing
- CNN --> Symbol Classification
- High character recognization accuracy, but lack capability to generate a complete LaTex sequence



Guillaume Genthial, Bolg: “Seq2Seq for LaTeX generation”

- Seq2Seq: Self-designed CNN Encoder - LSTM RNN Decoder with Attention layer
- Potential improvements: Well-defined model architectures; Positional encoding

$$
\begin{aligned}
h_t &= \text{LSTM}(h_{t-1}, [w_{t-1}, o_{t-1}]) \\
c_t &= \text{Attention}([e_1, \ldots, e_n], h_t) \\
o_t &= \tanh(W_3 \cdot [h_t, c_t]) \\
p_t &= \text{softmax}(W_4 \cdot o_t)
\end{aligned}
$$

BI-LSTM & Attention (Reasons of Approach)

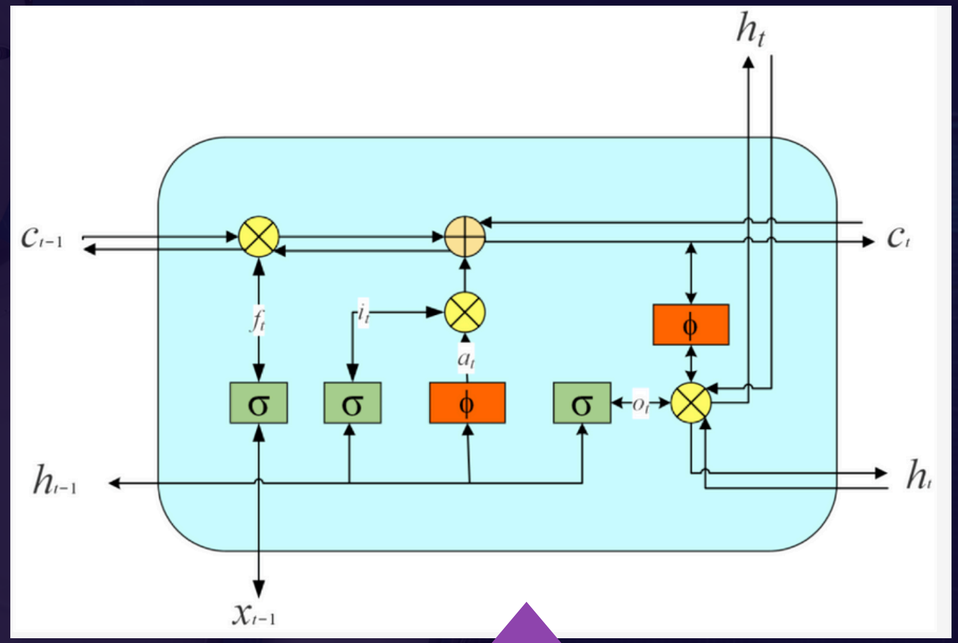
03

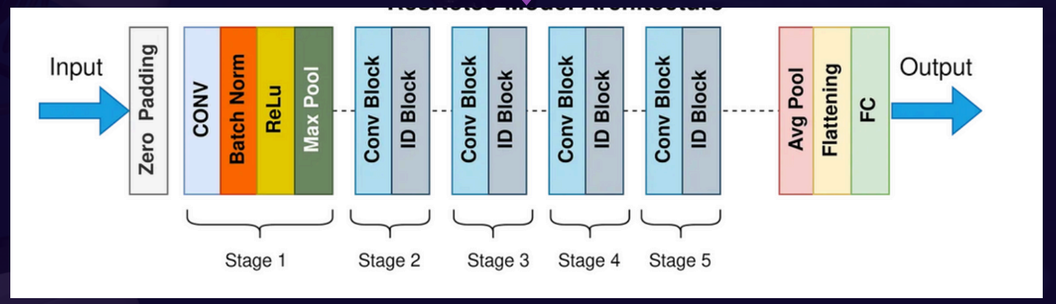
Contextual Understanding with BI-LSTM:

- **Bidirectional Processing:** Providing a understanding of the entire sequence. For mathematical equations, this means understanding how different parts of the equation relate to each other
- **Long-Term Dependencies:** LSTMs are designed to handle long-term dependencies in sequence data, where the relationship between components can be spread across the sequence.

Enhanced Focus with Attention Mechanism:

- **Selective Focus:** The relevance of different elements can vary greatly depending on the part of the equation being transcribed to LaTeX.
- **Improved Accuracy:** Ensuring that symbols and structures are correctly represented.





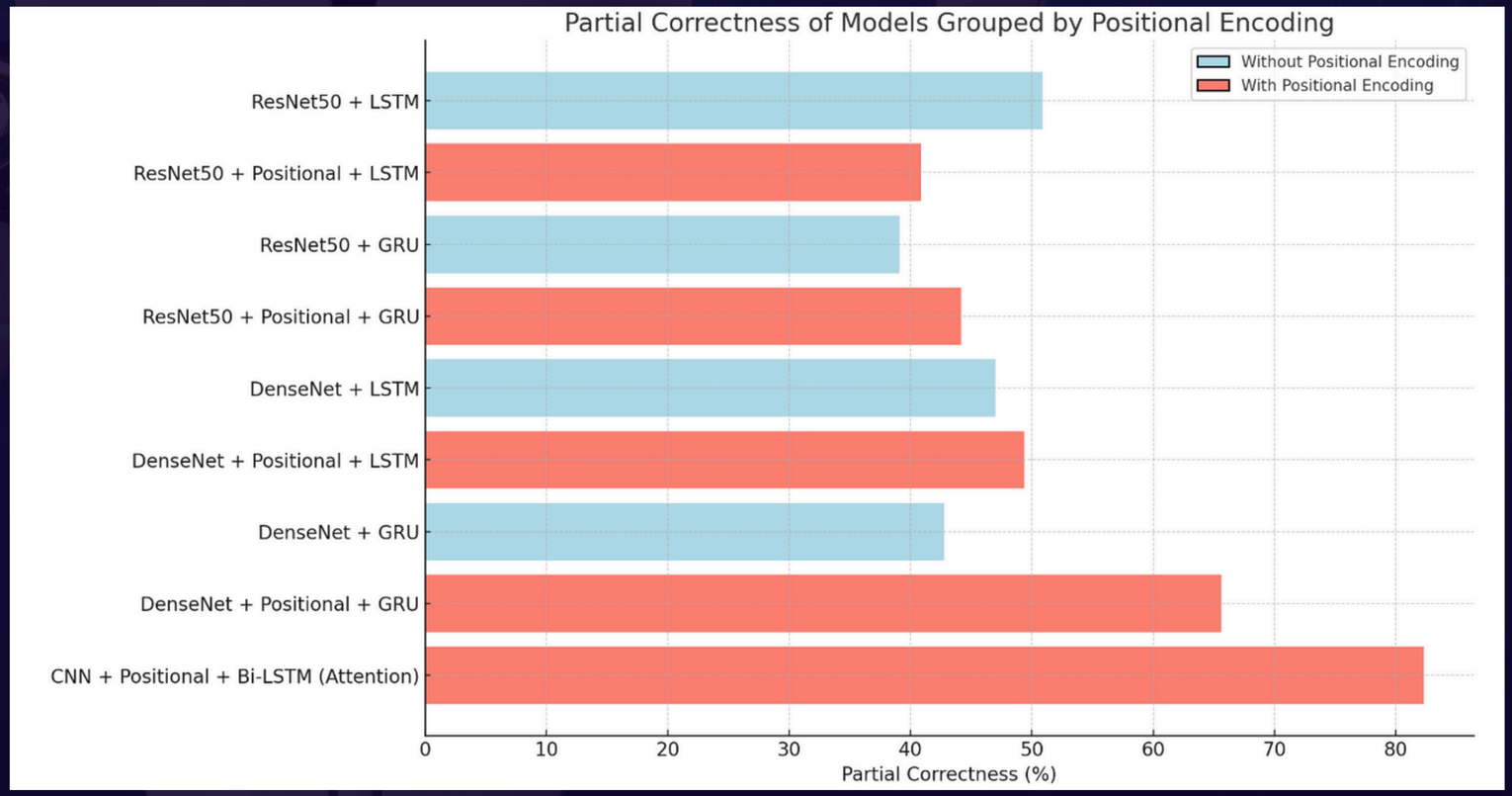
First Stage Result

- Partial Correctness: 60% of the tokens in the sequence match

Model	Partial Correctness
ResNet50 + LSTM	50.88%
ResNet50 + Positional + LSTM	40.90%
ResNet50 + GRU	39.12%
ResNet50 + Positional + GRU	44.20%
DenseNet + LSTM	47.00%
DenseNet + Positional + LSTM	49.40%
DenseNet + GRU	42.80%
DenseNet + Positional + GRU	65.60%
CNN + Positional + Bi-LSTM (Attention)	82.3%

Model Configs

"emb_dim": 80,
"max_len": 150,
"dropout": 0.0,
"cuda": True,
"batch_size": 16,
"epochs": 20,
"lr": 3e-4,
"min_lr": 3e-5,
"decay_k": 1.0
"lr_decay": 0.5,
"lr_patience": 3,
"clip": 2.0



- Simple CNN+RNN architectures are limited.
- Positional encoding helps.
- Attention mechanism significantly improves performance.

Implementation & Experiment

Models:
Encoder: **CNN**, Resnet 50, DenseNet
Positional Embedding: **True**/ False
DecoderCell: LSTM, GRU, **Bi-direction LSTM**
Optimizer: Adam
AttentionMechanism: Additive

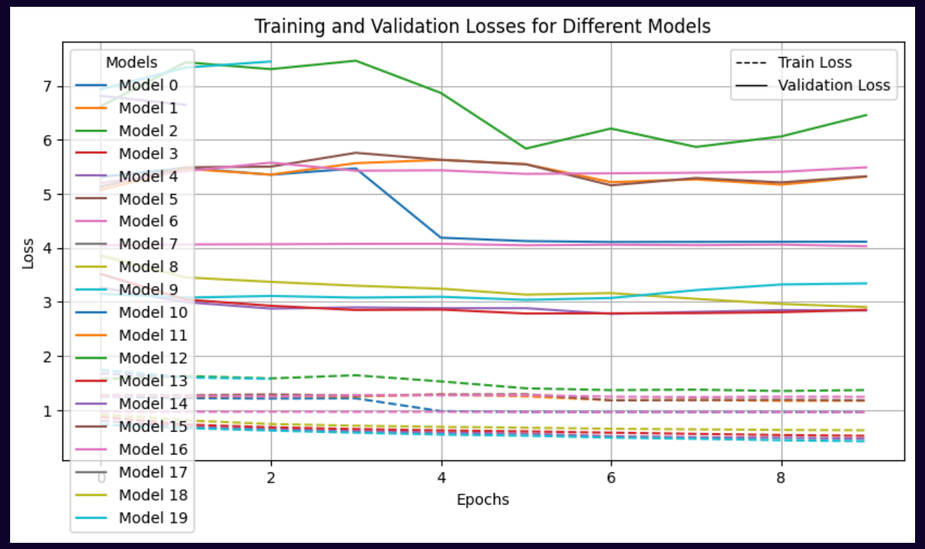
Hyperparameters:
n_epochs, batch_size: balancing between convergence speed & quality
dropout: to prevent overfitting
emb_dim: the dimensions for embedding
dec_rnn_h: the hidden layer of RNN
clip: gradient clipping, to prevent exploding gradient problem
lr, lr_init, lr_min: boundaries of the learning rate can adapt
decay, start_decay, end_decay: when to start and end decay
DecodingStrategy (for evaluation): Greedy Search/ Beam Search
Beam size: (call Greedy if =1)
Cell_config: num of hidden units, dim of embedding

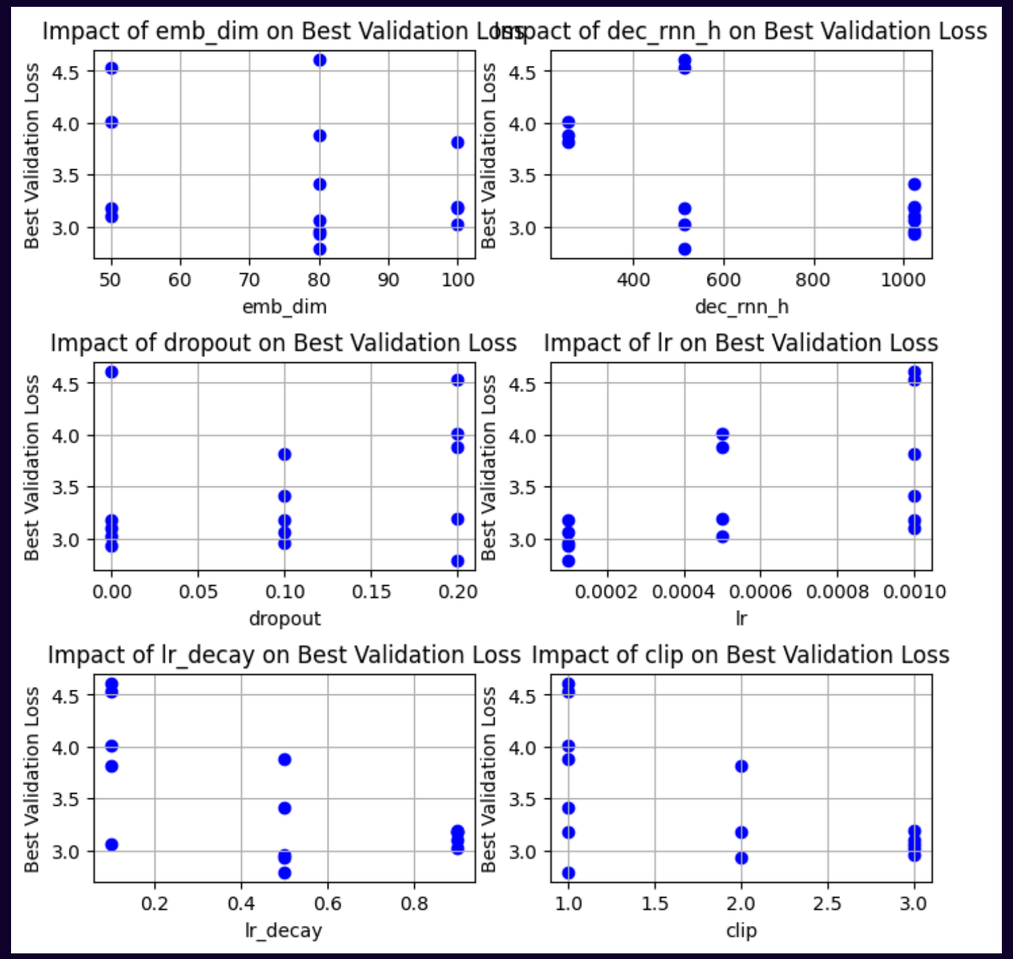
Hyper parameter tuning
Model Config: CNN w/ Positional Embedding + Bi-LSTM w/ Additive Attention

Random Search- on Trainer
"emb_dim": [50, 80, 100],
"dec_rnn_h": [256, 512, 1024],
"dropout": [0.0, 0.1, 0.2],
"lr": [5e-3,1e-3,5e-4, 1e-4],
"lr_decay": [0.1, 0.5, 0.9],
"lr_patience": [2, 3, 4],
"clip": [1.0, 2.0, 3.0]

Grid Search- on Latex Generator
“Beam size”: [1,3,5,10] (1 is Greedy Search)

Hyper Parameter Tuning on Trainer (Train)





'model tag': 4,
'hyperparameters':

emb_dim: 100
dec_rnn_h: 1024
add_position_features: True
dropout: 0.1
lr: 0.0001
lr_decay: 0.5
lr_patience: 2
clip: 2.0

Random Search

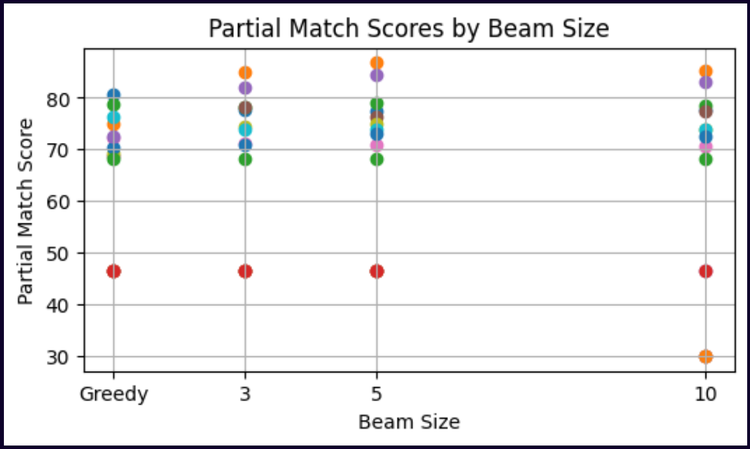
The initial learning rate should be low (0.0001)

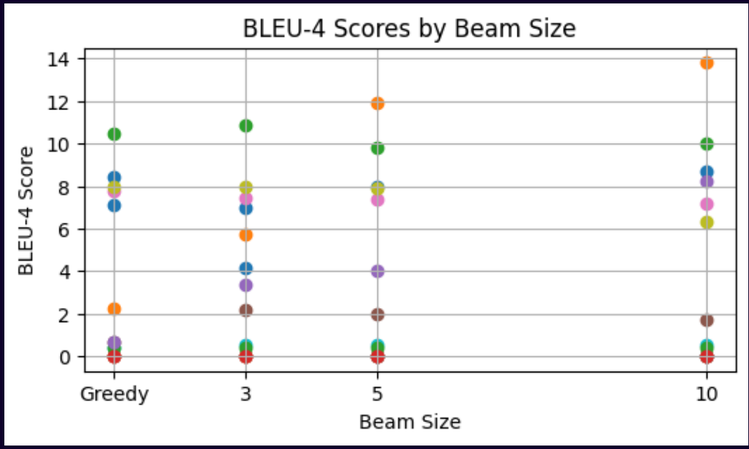
The best drop out rate is 0.1

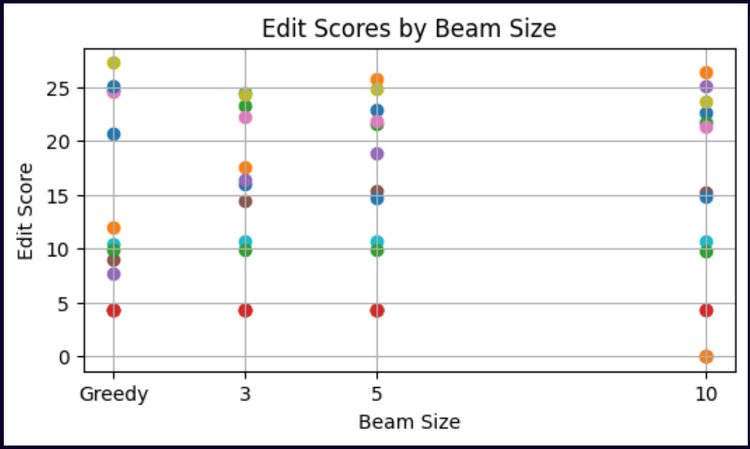
The clip should be higher to prevent gradient explode

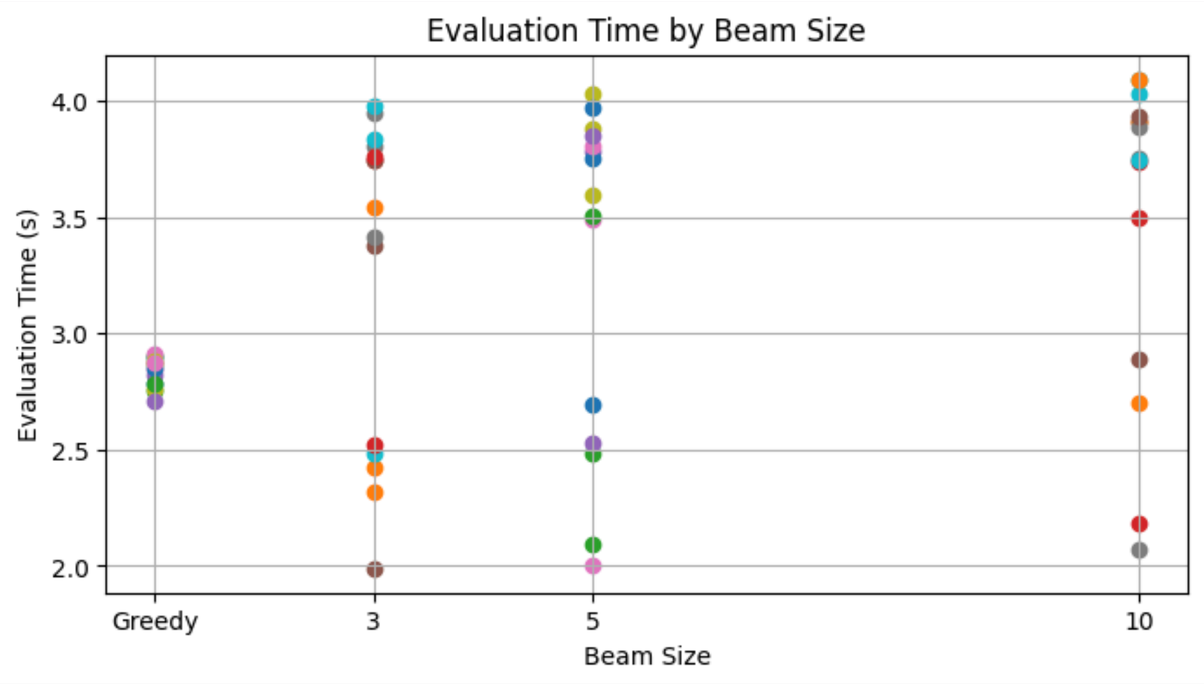
lr_decay should be higher to help convergence

Hyper Parameter Tuning on Latex Producer (Eval)



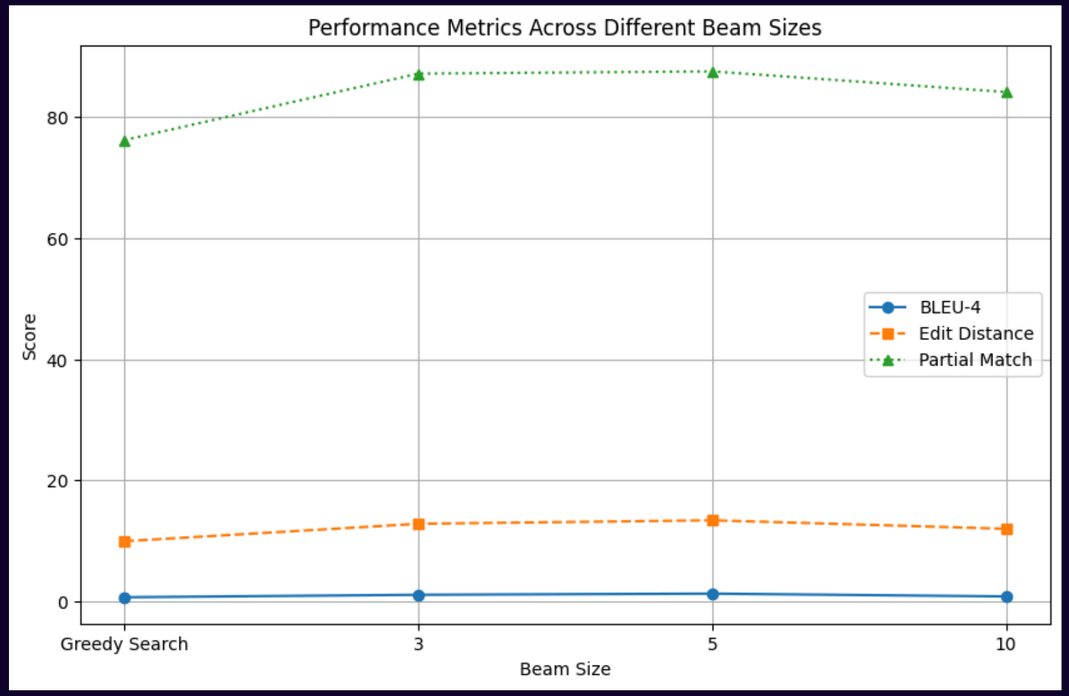






- **Beam Search generally out performs Greedy Search (Beam Size = 5 best)**
- **Beam requires more time and computation than Greedy Search**
- **The higher the beam size, the longer it takes**

The Best Model



'hyperparameters':
emb_dim: 100
dec_rnn_h: 1024
add_position_features: True
dropout: 0.1
lr: 0.0001
lr_decay: 0.4
lr_patience: 2
clip: 2.0

Performance w/ Beam size of 5 performs best:
'score':
{'BLEU-4': 1.2430632691273709,
'Edit': 13.361611876988333,
'Partial Match': 87.57575757575758}

Thank You!