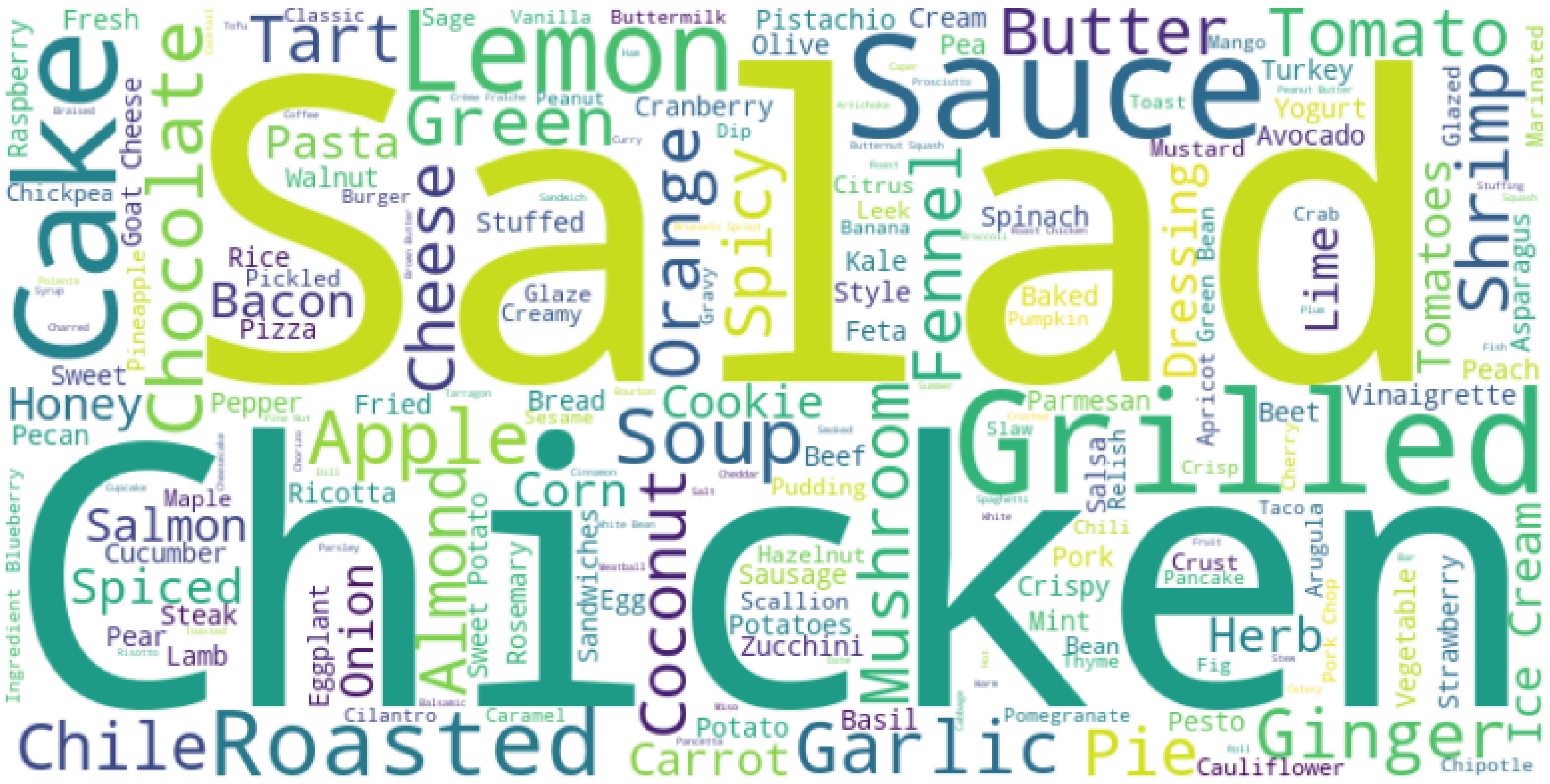
Session 3 - Image Captioning

Team 4:

- María José Millán
- Agustina Ghelfi
- Laila Aborizka

DATA EXPLORATION AND CLEANING

DATASET OVERVIEW: THE DATASET WAS FILTERED AS WE ONLY NEEDED (IMAGE NAME ,TITLE)



The word cloud shows the frequency of words in the titles, with "Salad" and "Chicken" appearing most frequently.

Shape: (13,501 rows, 2 columns)
Columns:
- Image Name : The filename of the image.
- Title : The caption or title describing the image.

Data Cleaning Steps:

Missing Values:
- Image Name: No missing values.
- Title: 5 missing values.

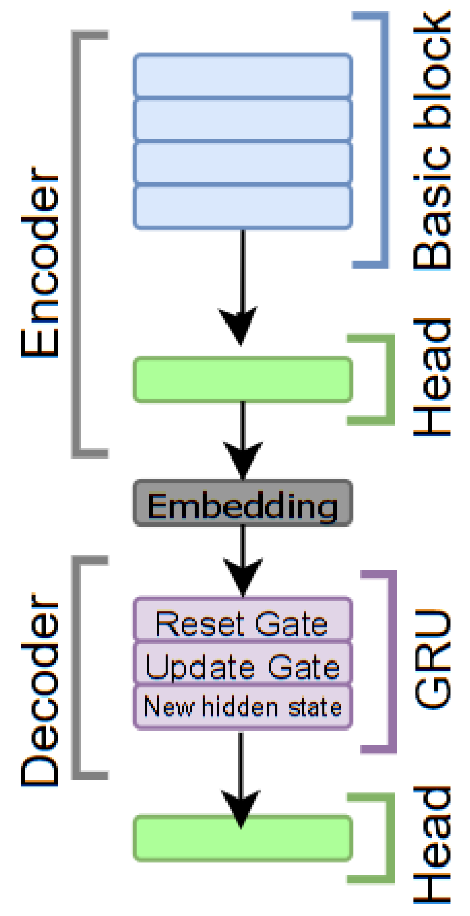
Cleaning Action:
- Titles with missing values were removed

Non-Existent Images:
- Some entries had Image Name values that didn't match any actual image files.
- These rows were removed to ensure the dataset only contains valid image-caption pairs.

BASELINE MODEL: RESNET-18 + GRU

To tackle the image captioning task, we used a baseline model consisting of a ResNet-18 encoder and a GRU decoder with character-level text representation. The model is trained end-to-end without freezing any layers. By using a character-level representation, the model learns to generate captions letter by letter instead of entire words, which can be beneficial for handling out-of-vocabulary terms but may also lead to longer sequence dependencies.

ARCHITECTURE OVERVIEW



Encoder

ResNet-18, pretrained on ImageNet, which processes the input image and extracts high-level features. The output of the encoder is a 512-dimensional feature vector obtained through global pooling.

Feature projection

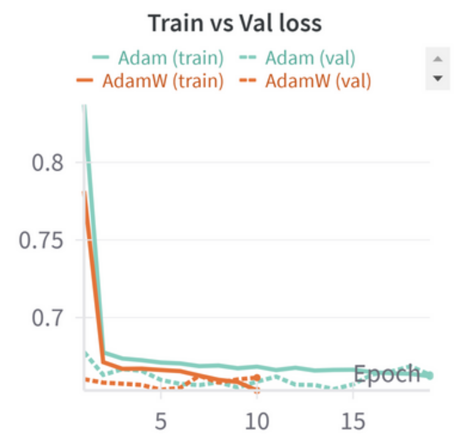
To adapt the visual features for caption generation, the model applies a linear projection layer that maps the 512-dimensional feature vector into the appropriate input space for the decoder, ensuring compatibility with the recurrent model.

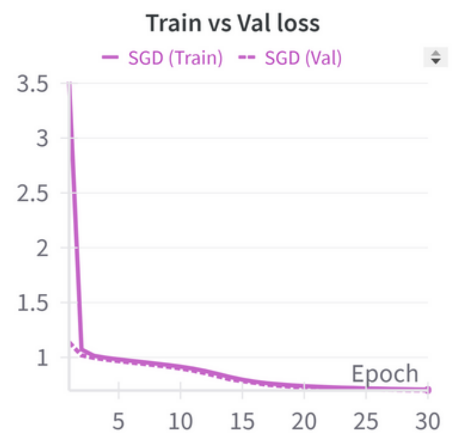
Decoder

Single-layer Gated Recurrent Unit (GRU) with 512 hidden units, which generates captions character by character. It receives an initial input token (<SOS>) and iteratively predicts the next character, using its own previous outputs as inputs until generating the <EOS> token.

TRAINING STRATEGY & EXPERIMENTS

To optimze the model's performance, several experiments were conducted, using Adam, AdamW and SGD optimizers, with learning rates between 1e-5, and 1e-2. Early stopping was applied with a patience of 5 and delta of 0.001.

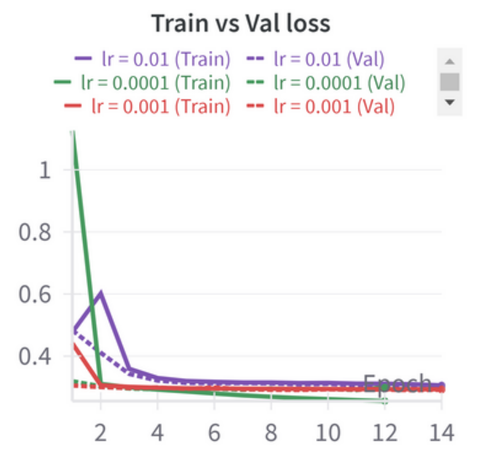




The curves illustrate how each optimizer affets convergence and overall performance for character-level model, with learning rate 1e-3.

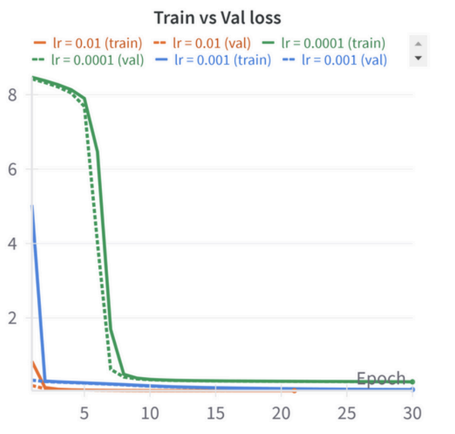
BEST: ADAMW

To further improve the model's performance we introduced word-piece and word level representations. For these, we also experimented with different hyperparameters.



Impact of varying the learning rate when using the word-piece-level model trained with the Adam optimizer.

BEST: LR = 0.001



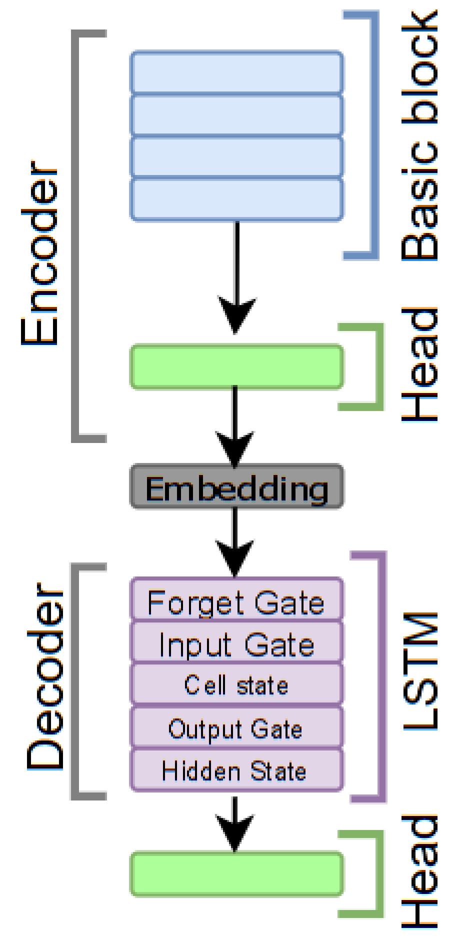
Impact of varying the learning rate when using the word-level model trained with the SGD optimizer.

BEST: LR = 0.01

RESNET-18 + LSTM

To improve the baseline model, we replaced the GRU decoder with a Long Short-Term Memory (LSTM) network, which is better suited for capturing long-range dependencies in sequential data. We tested this configuration with the three different text level representations (character-level, word-piece-level, and word-level).

ARCHITECTURE OVERVIEW



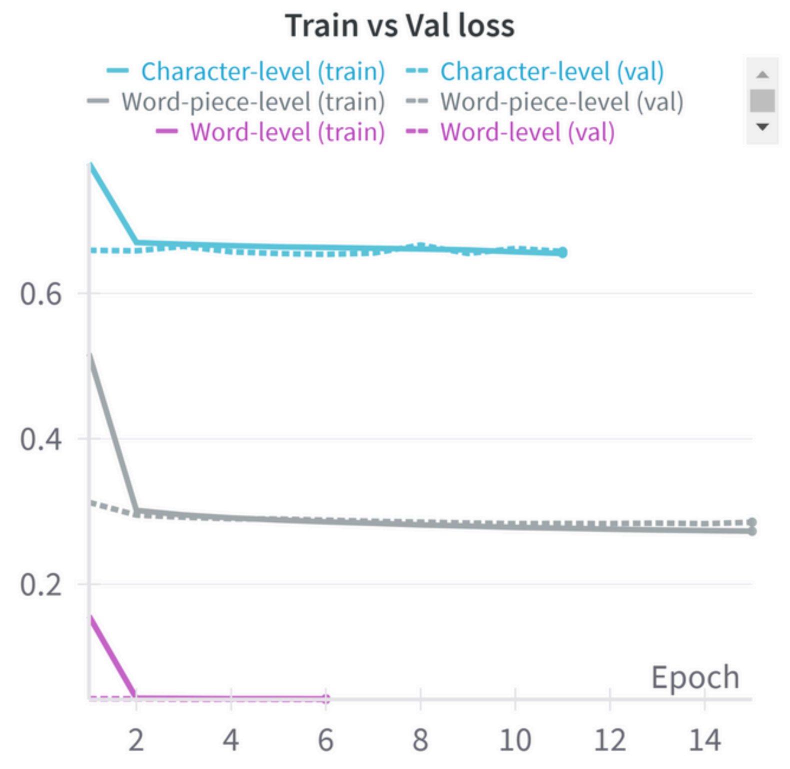
The LSTM decoder introduces an additional cell state, which helps preserve information over longer sequences, reducing the risk of vanishing gradients. Unlike GRU, which has a single hidden state, LSTM maintains two separate states: hidden state and cell state, allowing it to retain and discard information more selectively.

To integrate LSTM, the feature vector extracted by ResNet-18 is first projected to match the decoder's input space. The projected features are then used to initialize both the hidden and cell states of the LSTM. At each step, the decoder receives the previous character's embedding and predicts the next character until reaching the <EOS> token.

This modification was expected to improve caption quality by enhancing the model's ability to retain context over longer sequences, which is crucial for generating coherent and meaningful captions.

TRAINING STRATEGY & EXPERIMENTS

The figure below illustrates how training and validation loss vary when using different text representations while keeping all other parameters fixed (ResNet-18 + LSTM, AdamW optimizer, learning rate = 0.001, batch size = 32, dropout=0.3). The best performing model for each text representation was obtained with this setup.



Word-level representation achieves the lowest loss and converges the fastest, indicating that it is the most effective in this setup.

Word-piece-level generalizes well, but presents higher loss than word-level.

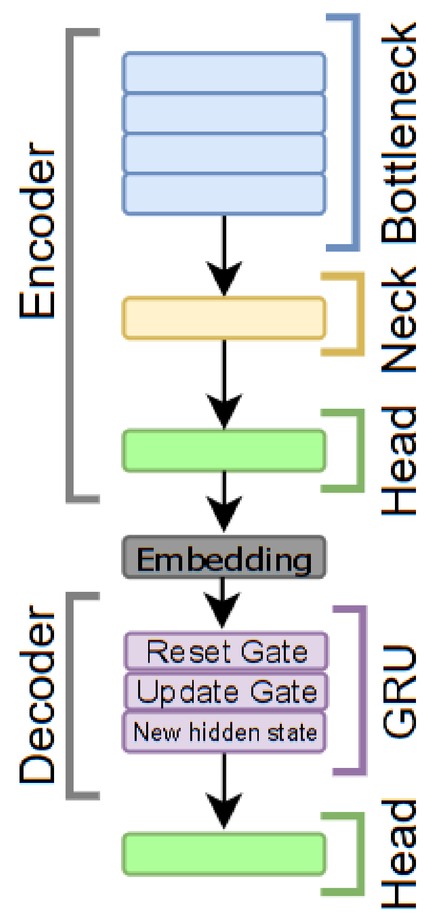
Character-level maintains the highest loss throughout training, likely due to the increased difficulty in capturing long-range dependencies, making optimization more challenging.

These results indicate that choosing the appropriate text representation has a significant impact on model performance.

RESNET-50 + GRU

To enhance the feature extraction capability of the model, we replaced the ResNet-18 encoder with a ResNet-50 network while keeping the GRU decoder unchanged. This modification aimed to provide richer and more detailed image representations, potentially leading to better caption generation.

ARCHITECTURE OVERVIEW



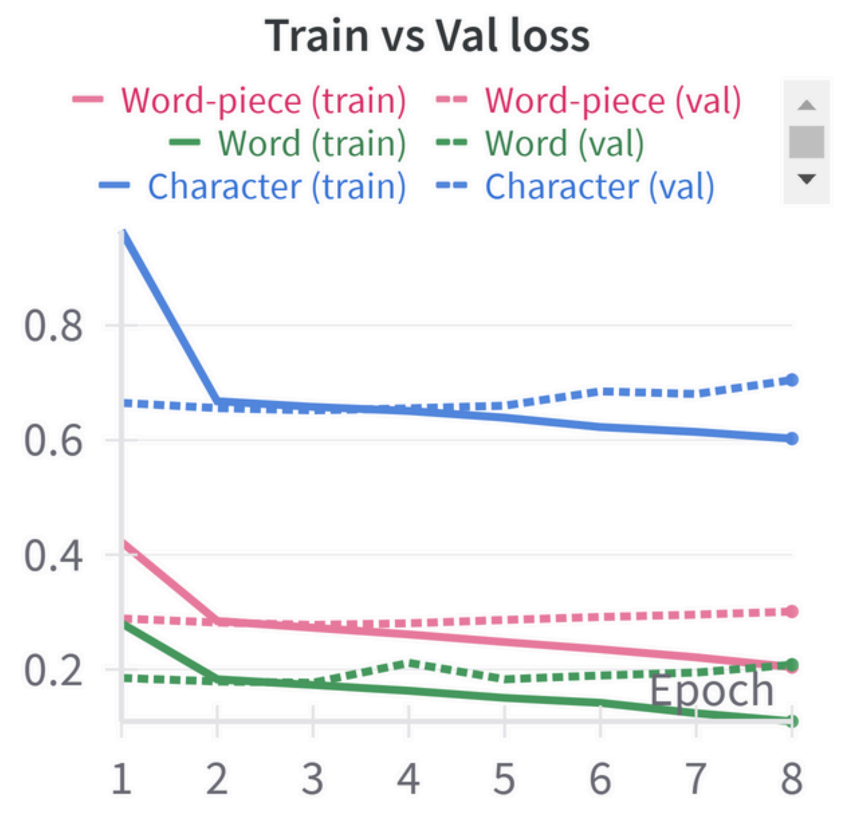
ResNet-50 is a deeper convolutional neural network compared to ResNet-18, consisting of 50 layers instead of 18. It incorporates bottleneck residual blocks, which allow for more efficient feature extraction without significantly increasing computational cost.

Since ResNet-50 outputs 2048-dimensional feature vectors (compared to 512 in ResNet-18), we added a linear projection layer to reduce the feature dimensionality from 2048 to 512, ensuring compatibility with the GRU decoder. This projection ensures that the decoder receives feature embeddings in the same space as in the baseline model.

By using ResNet-50, we expected the model to generate more descriptive captions, as the encoder would provide more refined and discriminative visual features, leading to improved text predictions from the decoder.

TRAINING STRATEGY & EXPERIMENTS

The figure below illustrates how training and validation loss vary when using different text representations while keeping all other parameters fixed (ResNet-50 + GRU, AdamW optimizer, learning rate = 0.001, batch size = 32). The best performing model for each text representation was obtained with this setup.



All three text representations exhibit overfitting (increasing gap between training and validation loss as training progresses).

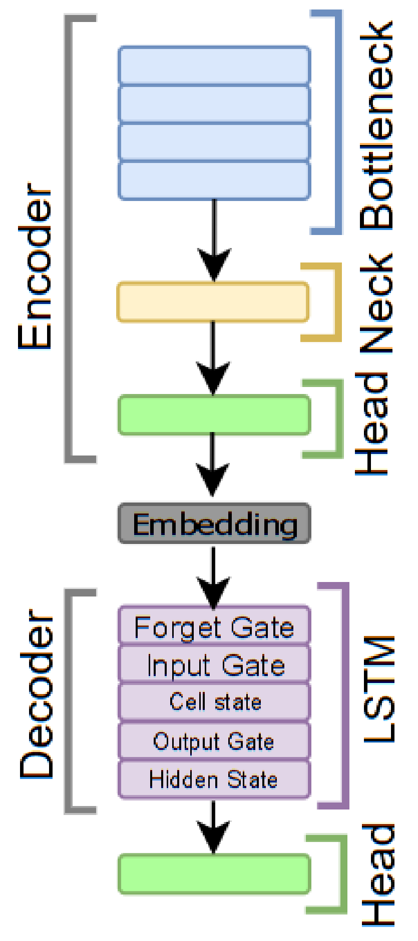
Interestingly, this overfitting behavior was not observed in the ResNet-18 + GRU setup. A possible explanation for this is the increased feature complexity in ResNet-50.

Since ResNet-50 extracts richer and more detailed features, the model might be memorizing training data instead of generalizing well to the validation set.

RESNET-50 + LSTM

For the final experiment, we combined the ResNet-50 encoder with an LSTM decoder, integrating the two modifications previously tested separately. This setup aimed to leverage the stronger feature extraction capabilities of ResNet-50 and the improved sequence modeling of LSTM, potentially leading to the best performance.

ARCHITECTURE OVERVIEW

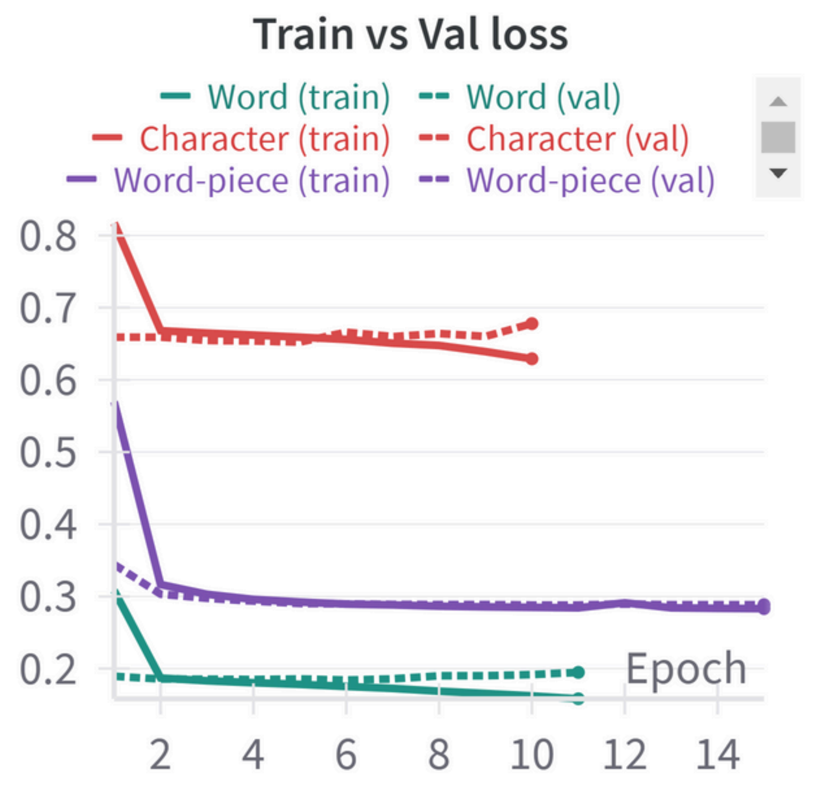


No additional modifications were needed beyond those previously applied when integrating ResNet-50 with GRU or ResNet-18 with LSTM. The same projection layer was used to map the 2048-dimensional ResNet-50 features to the 512-dimensional input space of the LSTM, ensuring compatibility.

By combining these two architectural changes, we expected the model to generate more accurate and descriptive captions, benefiting from the enhanced visual representations of ResNet-50 and the better memory retention of LSTM.

TRAINING STRATEGY & EXPERIMENTS

The figure below illustrates how training and validation loss vary when using different text representations while keeping all other parameters fixed (ResNet-50 + GRU, AdamW optimizer, learning rate = 0.001, batch size = 32). The best performing model for each text representation was obtained with this setup.



The loss curves for the ResNet-50 + LSTM setup indicate different behavior compared to the previous experiments. While some overfitting is still present (especially for character and word-based representations), it appears to be less pronounced than in the ResNet-50 + GRU setup. This suggests that LSTM's better memory retention helps mitigate overfitting to some extent.

The combination of ResNet-50 + LSTM provides a balance between improved feature extraction and effective sequence modeling. However, word-based representations remain more prone to overfitting, suggesting that careful regularization is still needed.

QUANTITATIVE RESULTS

BLEU-1:
Measures the precision of unigrams (single words) in the generated captions compared to the reference captions. It provides an indication of how many words match but does not consider sentence structure.

ROUGE-1:
Evaluates the recall of unigrams, capturing how much of the reference text is covered by the generated captions. It is useful for assessing content overlap.

METEOR:
Evaluates both precision and recall, while also considering synonyms and word order. This makes it a more comprehensive metric for assessing the quality of generated captions compared to the reference text.

BASELINE MODEL: RESNET-18 + GRU

	CHAR	**WORD**	WORDPIECE
BLEU-1	0	**0.071**	0.036
ROUGE-1	0	**0.120**	0.056
METEOR	0	**0.047**	0.034

RESNET-50 + GRU

	CHAR	**WORD**	WORDPIECE
BLEU-1	0	**0.101**	0.068
ROUGE-1	0	**0.127**	0.086
METEOR	0	**0.055**	0.047

RESNET-18 + LSTM

	CHAR	**WORD**	WORDPIECE
BLEU-1	9.7e-5	**0.126**	0.045
ROUGE-1	0.001	**0.159**	0.065
METEOR	5.6e-5	**0.062**	0.037

RESNET-50 + LSTM

	CHAR	**WORD**	WORDPIECE
BLEU-1	6.4e-5	**0.087**	0.056
ROUGE-1	0.0001	**0.107**	0.070
METEOR	5.7e-5	**0.044**	0.038

Highlighted with **bold**: best text-level representation for each model.
Highlighted with ▭: best model for each text-level representation.

ANALYSIS & CONCLUSIONS

The results highlight that none of the models achieved strong performance overall, indicating that the task remains challenging under the tested configurations. ResNet-18 + LSTM with word-based representation performed the best across all metrics, suggesting that the combination of a simpler visual encoder and an LSTM decoder was the most effective for caption generation.

ResNet-18 + LSTM achieved the highest scores, reinforcing that increasing the encoder's depth (ResNet-50) did not necessarily translate to better textual outputs.

Word-piece and character-based representations performed poorly in all cases, indicating that sub-word and character-level tokenization might not be well-suited for this task, at least with the tested architectures.

QUALITATIVE RESULTS

CHARACTER LEVEL

Groundtruth: Creole Cream Cheesecake With Caramel-Apple Topping
Prediction: Caaeee   e

Groundtruth: Saffron Pilaf
Prediction: Saaeee   e

WORD LEVEL

Groundtruth: Our Favorite Chocolate Chip Cookies
Prediction: Chocolate Chocolate

Groundtruth: Ultimate grilled sandwich
Prediction: grilled and with

WORDPIECE LEVEL

Groundtruth: lemon icebox pie
Prediction: lemon -

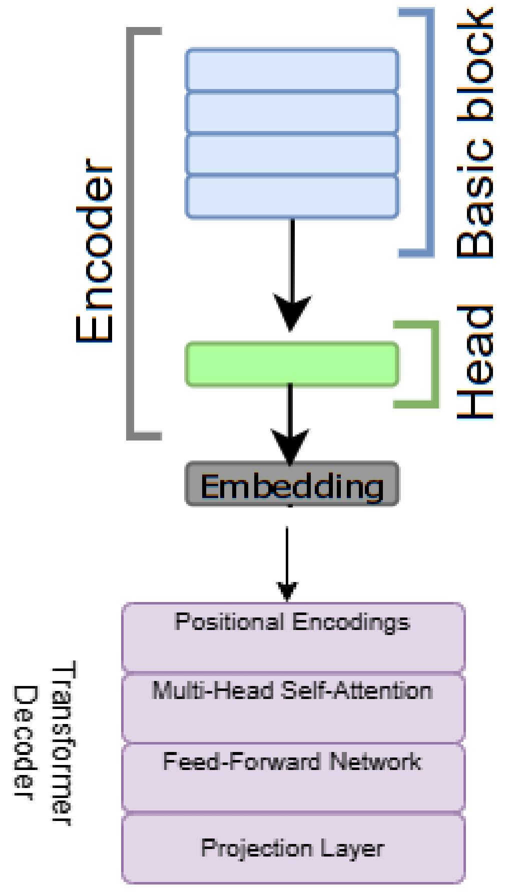
Groundtruth: breakfast pizza with sausage, eggs, spinach, and cream
Prediction: grilled with with with

Consistent with our quantitative results, the generated captions were not accurate. However, in the word-level approach, some words, like "chocolate", were correctly predicted. In the WordPiece approach, when the ground truth contained many ingredients, the model failed to predict them but seemed to recognize their presence, repeatedly using "with", as in "with with with". This suggests some structural learning despite lexical inaccuracies.

RESNET-18 + TRANSFORMER DECODER

To improve the baseline model, we replaced the GRU decoder with a Transformer Decoder , hypothesizing that multi-head attention mechanism in Transformers can focus on different parts of the input sequence simultaneously, potentially leading to richer contextual representations and more accurate captions. The rest of the architecture, including the ResNet-18 encoder and character-level text representation, remained unchanged

ARCHITECTURE OVERVIEW

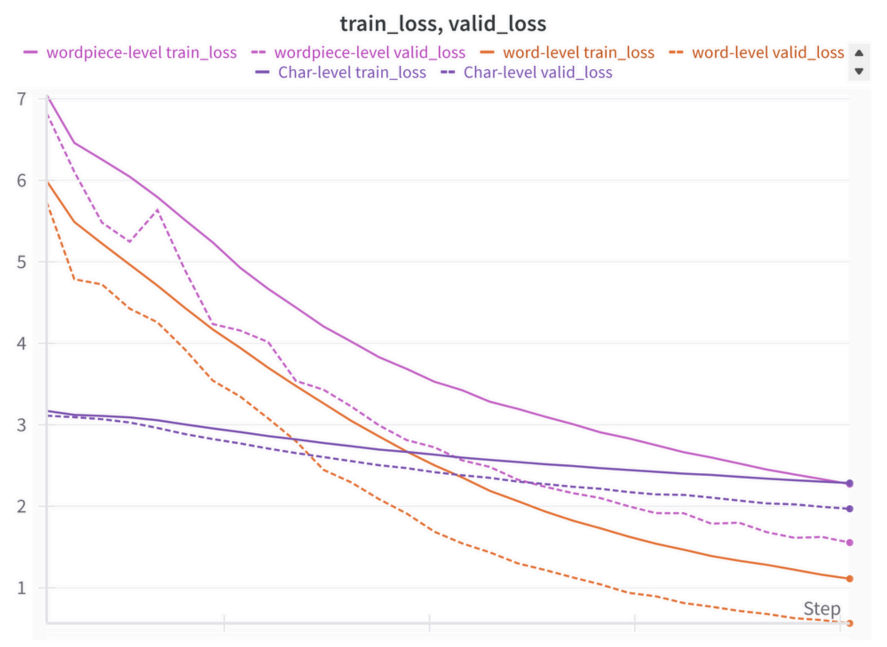


The Transformer decoder in this model is built using PyTorch's nn.TransformerDecoder module, which is made up of multiple nn.TransformerDecoderLayer layers. Each layer has multi-head self-attention and feed-forward networks, helping the model generate captions by considering both the visual features from the ResNet encoder and the previously generated tokens.

The multi-head attention allows the decoder to focus on different parts of the input at once, capturing long-range dependencies and relationships. The decoder also uses positional encodings to maintain the order of tokens, ensuring the model understands the sequence of the text.

TRAINING STRATEGY & EXPERIMENTS

- num_layers = 1 (number of Transformer decoder layers).
- nhead = 2 (number of attention heads in Transformer).
- dim_feedforward = 512 (feedforward layer dimension in Transformer).
- dropout = 0.1 (dropout rate in Transformer).
- batch_size = 8
- learning_rate = 1e-4 (learning rate for Adam optimizer).
- epochs = 30



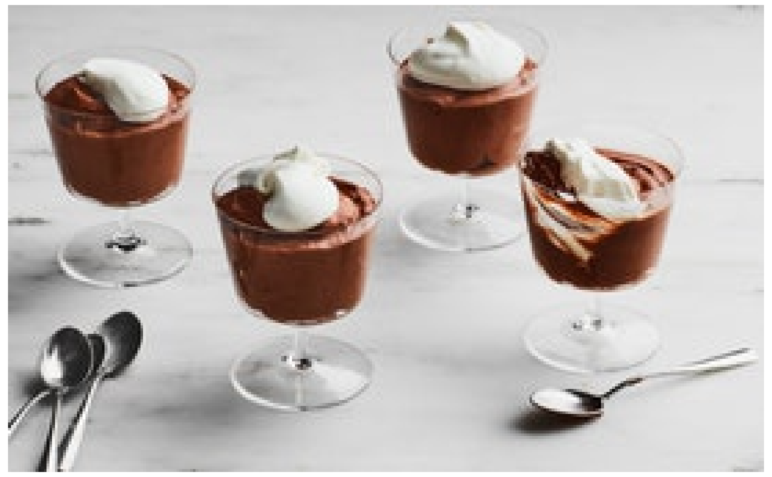
QUANTITATIVE RESULTS ON TEST SET

	CHAR	WORD	WORDPIECE
BLEU	0.082	0.150	0.72
ROUGE	0.150	0.115	0.706
METEOR	0.123	0.113	0.711

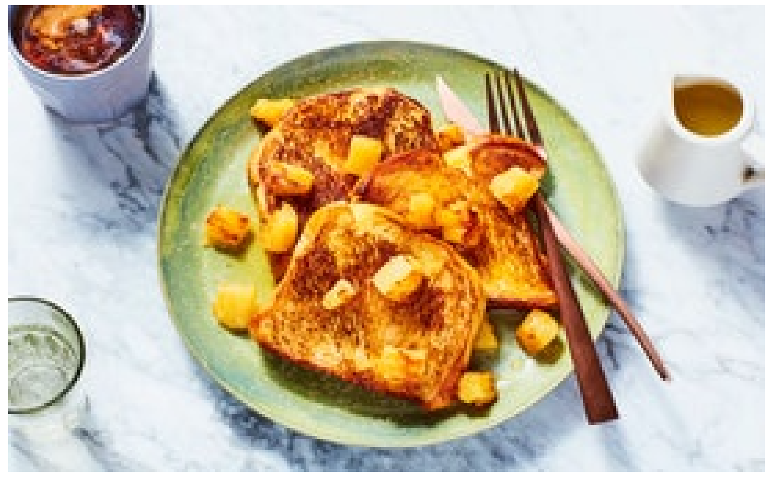
Word-level representation achieves the lowest loss and converges the fastest, highlighting that it is the most efficient however it didn't perform the best on the Test set..

QUALITATIVE RESULTS - TRANSFORMER DECODER

WORD LEVEL



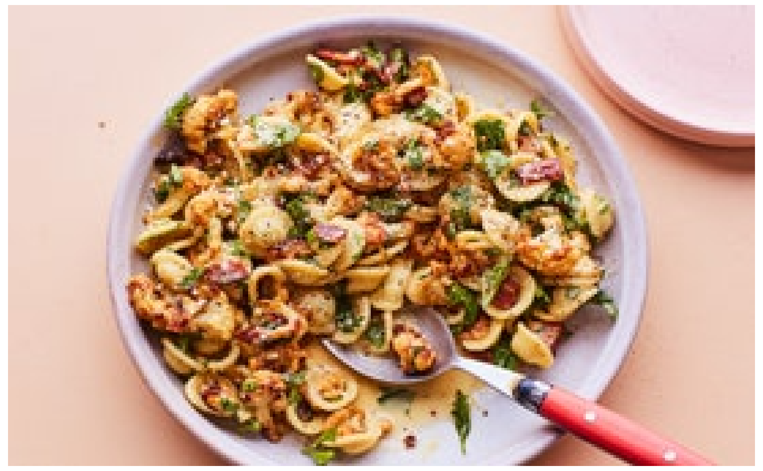
GROUND TRUTH: CHOCOLATE MOUSSE
PREDICTED: MOUSSE MOUSSE MOUSSE MOUSSE



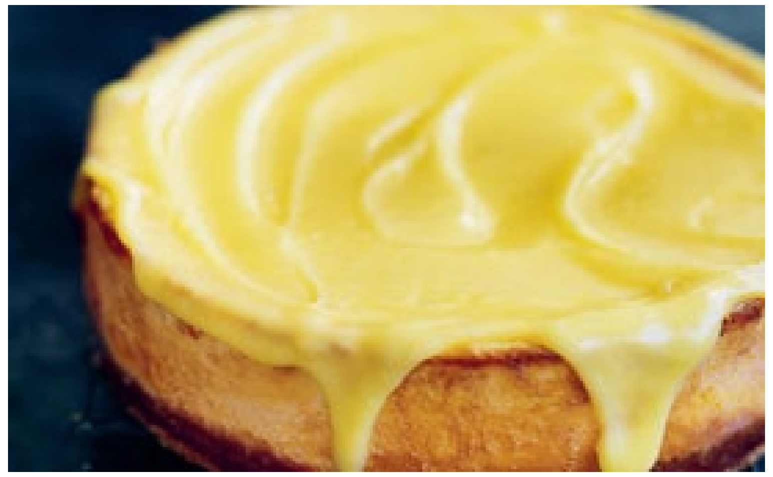
GROUND TRUTH: COCONUT MILK FRENCH TOAST WITH PINEAPPLE SYRUP
PREDICTED: COCONUT COCONUT MILK CAKE WITH PINEAPPLE

The model captures key elements like ingredients but struggles with text repetition

CHAR LEVEL



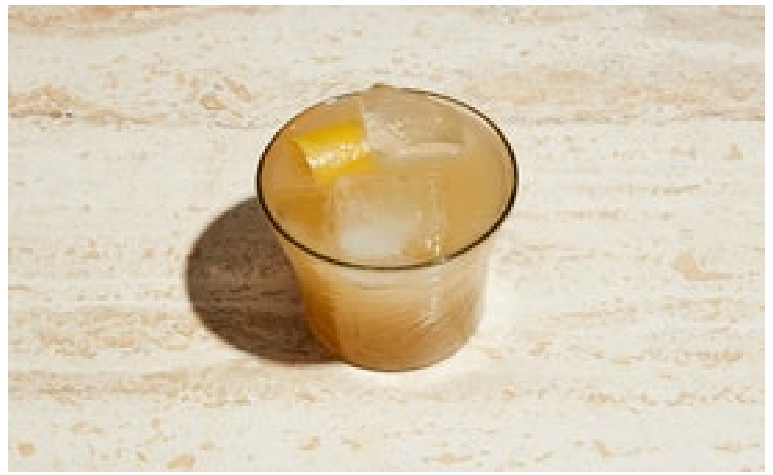
GROUND TRUTH: CAULIFLOWER CARBONARA
PREDICTED: CAULIFLOWER CULLEECRLE



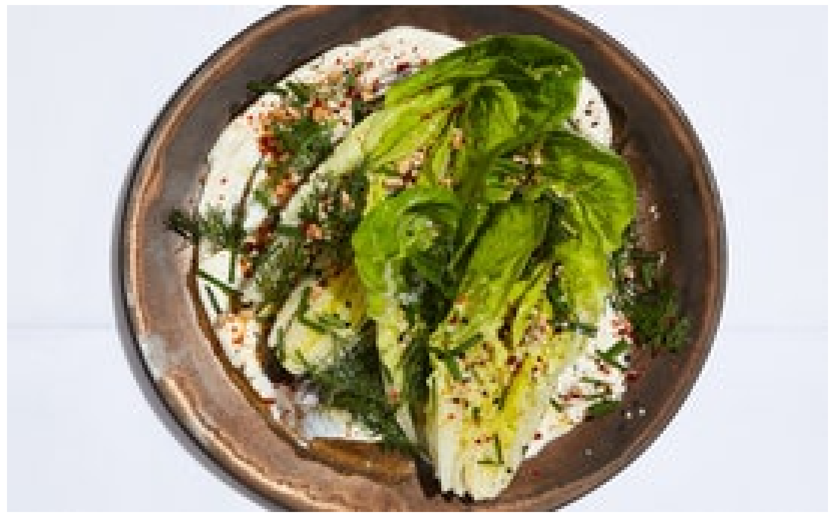
GROUND TRUTH: LEMON CHEESECAKE
PREDICTED: LEMON CHKEEECEKEE

The model could capture some words correctly but it produced a lot of nonsensical text with repetitive characters

WORDPIECE-LEVEL



GROUND TRUTH: CYNAR SPRITZ
PREDICTED: CYNAR SPRITZZ



GROUND TRUTH: LITTLE GEM WEDGE SALAD WITH TAHINI RANCH
PREDICTED: LITTLE GEM WEDGE SALAD TA TA TA RANCHHINIHINI

The model's predictions show a mix of partial correctness and some repetitive or nonsensical outputs

TEACHER FORCING

Teacher Forcing is a training technique used in the decoder of sequence models. Instead of using the model's own predictions in every input for the next step, we sometimes provide the ground truth sequence from the dataset, using the correct token from the training data.

- Standard Approach: The model generates an output and feeds it into the next step.
- Teacher Forcing: The ground truth token is used as input at each step, guiding the model.

Hypothesis: By applying Teacher Forcing in the decoder, the training will be more stable and efficient. Also, the performance will improve.

EXPERIMENTS

We applied teacher forcing technique to our best models for each text-level representation. For character and word level representations no further improvements were achieved but for word-piece representation there was a slight improvement across all metrics.

RESNET-50 + GRU - WORD-PIECE LEVEL

	Previous predictions	**Teach forcing predictions**
BLEU-1	0.068	**0.127**
ROUGE-1	0.086	**0.106**
METEOR	0.047	**0.063**

Groundtruth: lemon icebox pie
Previous prediction: lemon -
Teach forcing prediction: lemon - pie

SUMMARY

MODEL	BEST REPRESENTATION	BLEU-1	ROUGE-1	METEOR
ResNet-18 + GRU	WORD	0.071	0.120	0.047
ResNet-18 + LSTM	WORD	0.101	0.127	0.055
ResNet-18 + Transformer	WORD-PIECE (Best Overall)	**0.72**	**0.706**	**0.711**
ResNet-50 + GRU	WORD	0.126	0.159	0.062
ResNet-50 + LSTM	WORD	0.087	0.107	0.044
Teacher forcing: Resnet-50 + GRU	WORD-PIECE	0.127	0.106	0.063

GROUND TRUTH: CYNAR SPRITZ
PREDICTED: CYNAR SPRITZZ

GROUND TRUTH: LITTLE GEM WEDGE SALAD WITH TAHINI RANCH
PREDICTED: LITTLE GEM WEDGE SALAD TA TA TA RANCHHINIHINI

KEY OBSERVATIONS

- ResNet-18 + LSTM achieved scores higher than ResNet-50 + LSTM , reinforcing that increasing the encoder's depth did not necessarily translate to better textual outputs. However with GRU it was the other way around.
- Generally character level representation performed the worst.
- Switching from GRU and LSTM to a transformer significantly improved performance, highlighting the positive impact of attention. By considering both visual features and predicted tokens, the model better predicts the next token.

TEACHER FORCING

Teacher Forcing is a training technique used in the decoder of sequence models. Instead of using the model's own predictions in every input for the next step, we sometimes provide the ground truth sequence from the dataset, using the correct token from the training data.

WORD-PIECE LEVEL

Groundtruth: lemon icebox pie
Previous prediction: lemon -
Teach forcing prediction: lemon - pie