CS6140 Final Report: Comparing Image Captioning Results Using RNNs, RNN-LSTMs, and Transformers

Trevor Powers POWERS.TR@NORTHEASTERN.EDU
Khoury College of Computer Science

James McKee MCKEE.JA@NORTHEASTERN.EDU
Khoury College of Computer Science

Nisarg Patel PATEL.NISARGS@NORTHEASTERN.EDU
Khoury College of Computer Science

Editor: Powers, McKee and Patel

Keywords: Image Captioning, RNN, LSTM, CNN, Transformer

1. Introduction

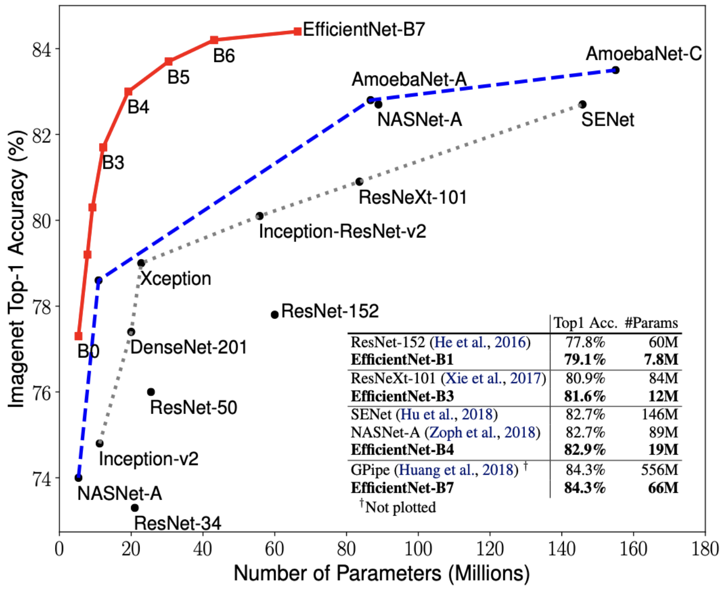
Image captioning is the process of automatically describing what is observed in the image. This relatively new field of machine learning lends itself to a variety of applications, from virtual assistants to the development of tools for the disabled. It involves the combination of domains including Computer Vision and Natural Language Processing. The computer vision part encodes the features of the image and those features are fed into the natural language processing part which decodes the image into a sequence of text. There has been a recent increase in popularity of the image captioning problem partially due to recent progress in object detection, attribute classification, action recognition, etc., and partially due to the increased availability of datasets. The Microsoft led Common Objects in Context (COCO) Image Captioning challenge provided a large open-source dataset of over 330,000 images with 5 captions per image which became the current state of the art dataset for the problem.(Lin et al. (2014))

In this project we compared 3 different machine learning decoder architectures to predict natural language words from the encoded features of the image. The first of the models uses a vanilla Recurrent Neural Network that cannot keep track of arbitrary long-term dependencies in the input sequences. The second model uses a special type of RNN known as Long Short Term Memory which partially solves this problem. Using the RNN and LSTM still has the issue of sequential processing. The tokens generated from the input captions need to be processed individually, thus for a sentence of n tokens, it would require n states of the sentences and each state is dependent on its previous state making it difficult to parallelize and highly inefficient.

Our third model uses Transformers to address these problems with the Attention mechanism. Attention mechanisms let a model draw from the state at any preceding point along the sequence. The attention layer can access all previous states and weight them according to a learned measure of relevance, providing relevant information about far-away tokens. Transformers use an attention mechanism without an RNN, processing all tokens at the same time and calculating attention weights between them in successive layers. Since the attention mechanism only uses information about other tokens from lower layers, it can be computed for all tokens in parallel, which leads to improved training speed. We compared the performance of the models using BLEU-2, BLEU-3, METEOR and ROUGE-1 which are some of the widely used evaluation metrics for image captioning.

2. Dataset

This project makes use of the Microsoft COCO: Common Objects in Context Dataset **?**. Particularly, we use the image caption portion of the dataset. To our knowledge, this dataset is currently the largest open-source image captioned dataset at the time of writing. It consists of approximately 328 thousand images. Microsoft states that each image has five written captions describing what is occurring in the image assigned to it. However, in our experience with the 2017 version of the caption dataset, not all images had five captions. We removed images that had less than 5 captions to maintain uniformity and reduce picture-to-picture bias across our training set. We also did not make use of all images, in order to reduce computational power in our training phase. Ultimately, we used 80,067 images in training and 20,017 in testing.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) [†]	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

[†]Not plotted

Figure 1: EffecientNet performace on ImageNet Relative to other state-of-the-art models

3. Methods

As stated above, the task of image captioning has historically been approached by utilizing an encoder and decoder. The encoder takes in an image, and encodes it within some higher dimensional feature space. The output of the encoder is then passed to the decoder. The decoder then outputs a caption attempting to describe what is occurring in the image. Finally, for validation of these models, we must explore various validation metrics. The following section discusses our choices of encoder and decoder implementation, as well as three validation metrics we employ to test our models.

3.1 The Encoding Network: EfficientNetB0

In this paper, we choose to use a convolutional neural network (CNN) to encode our images. CNNs are state-of-the-art for encoding images in machine learning tasks due to their use of convolutions to preserve spatial relationships in multi-dimensional data. In particular, we make use of EfficientNet to encode our input images as in Tan and Le (2019). EfficientNet was created by Google AI in 2019 and is currently considered state-of-the-art for image recognition tasks. EfficientNet outperforms other popular current models on ImageNet (an image database by WorldNet often used as a baseline for image recognition models), see Figure 1.

EfficientNet's superior performance is due to its use of compound model scaling. Particularly, EfficientNet optimizes model performance by selecting model depth, width, and resolution, subject to the equations (where ϕ is used to uniformly scale all other terms):

$$\text{depth: } d = \alpha^{\phi},\ \text{width: } w = \beta^{\phi},\ \text{resolution: } r = \gamma^{\phi}$$

$$\text{s.t. } \alpha * \beta^2 * \gamma^2 \approx 2, \alpha \geq 1, \beta \geq 1, \gamma \geq 1,$$

Thus, EfficientNet provides us with a predetermined network architecture that we use in this project. Particularly, we select EfficientNetB0, which has the lowest ImageNet accuracy (See Figure 1) but also the lowest number of parameters, as we chose to sacrifice accuracy for less compute time, given the short timeline of this project.

3.2 Decoding Network 1: RNN

The simplest decoder uses a recurrent neural network (RNN) to convert the image features to a text vector. Specifically it takes the image features and passes them through a fully connected feed-forward layer. It also takes the caption as a sequence, passes it through an embedding layer, and then passes it through the fully recurrent layer. Finally the image features and the embedded sequence are passed together through another fully connected layer.

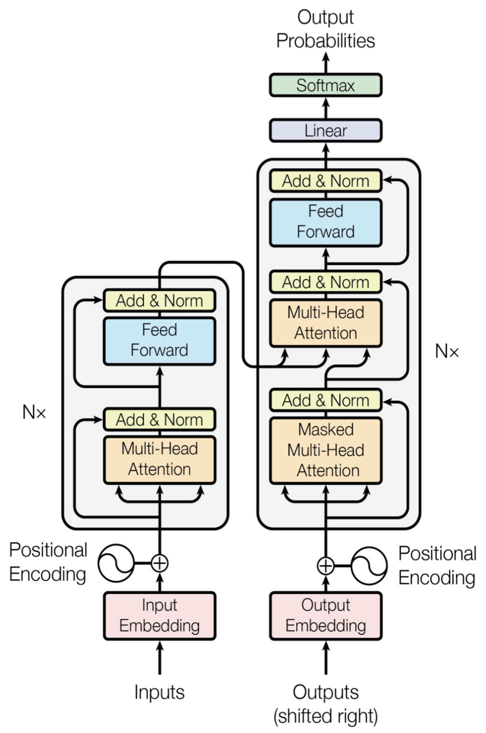


Figure 2: Attention Is All You Need Transformer Architecture

3.3 Decoding Network 2: LSTM

The second decoder model uses a special type of Recurrent Neural Network known as Long Short-Term Memory(Hochreiter and Schmidhuber (1997)). Unlike standard feedforward neural networks, LSTM has feedback connections. This allows LSTM to be capable of learning long-term dependencies which partially solves the vanishing gradient problem. Like RNN, LSTM also has chain structure. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate.

The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. The forget gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. The output of the input gate will decide which information is important to keep. The output gate decides what the next hidden state should be. The hidden state stores information of previous sequences. The new cell state and hidden state is then carried over to the next time step.

3.4 Decoding Network 3: Transformer

For our final decoder, we utilize a transformer. A transformer is a deep-learning model recently developed by Google Brain in 2017 Vaswani et al. (2017). It relies on self-attention, allowing the model to describe dependencies and relationships that exist between tokens of the input sequence, or in our case, text. Following the development of transformers, transformers have quickly become state-of-the-art for many Natural Language Processing tasks. Broadly, transformers depend on an encoder to map input sequences to a sequence of continuous representations. These representations are then fed to the decoder, alongside the output.

In this project, we make use of the Keras.io code base for their image captioning model Keras.io. This model follows the architecture outlined in Attention is All You Need, shown in Figure 2. Particularly, the input to the encoder is the output of EfficientNetB0 run on a given image. We then pass this output, alongside our image captions, into the decoder. We calculate the loss values used to update the trainable model weights based on the predictions outputted by the decoder with respect to each of the five captions assigned to the input image. In regards to the relationship between Figure 2 and the Keras code base, both the encoder and decoder models' input are initially run on the Positional Embedding Keras object. Then, the encoder layer normalizes the input, runs it through a dense activation layer, then through a self-attention layer of a MultiHeadAttention Keras object, and finally a normalization layer. This output is passed to the decoder. The decoder runs its input through a normalization layer followed by a self-attention layer of a MultiHeadAttention Keras object twice (the first time uses Masked Multi-Head Attention), before running it through a series of feed-forward networks and dropout layers. It then outputs its predictions. Note that we chose to use 8 attention heads for the encoder and 16 for the decoder to balance computational expense with model performance.

3.5 Metrics

BLEU: BiLingual Evaluation Understudy (BLEU) was proposed in 2002 for the evaluation of machine translation tasks by Papineni et al. (2002). It utilizes what the authors call 'modified n-gram precision.' For each n, the n-gram precision score is calculated by summing the number of n-grams in the predicted string which also appear in the reference string, with each n-gram count limited by the count in the reference, and dividing by the number of n-grams in the prediction. This calculation favors smaller predictions, so to balance this bias, there is a brevity penalty for predictions that are the same length or shorter than the reference string. Weights are also included for each n-gram order. For a prediction of length c, a reference of length r, and a precision p with weight w:

$$BLEU = BP * e^{(\sum_n w_n log(p_n))} \quad where \;\; BP = \begin{cases} e^{(1-\frac{r}{c})} & \text{if } c \leq r \\ 1 & \text{otherwise} \end{cases}$$

We have measured two different BLEU values. One with an equal weighting of 1-gram and 2-gram scores, and one with an equal weighting of 1-gram, 2-gram, and 3-gram scores. We did not include any 4-gram metrics.

METEOR: Metric for Evaluation of Translation with Explicit ORdering (METEOR), like BLEU, is a machine translation evaluation algorithm based on n-gram matching from Banerjee and Lavie (2005). METEOR was developed in response to the perceived weakness of BLEU, including the lack of recall, use of higher order n-grams, and geometric averaging of n-gram scores. The comparison of two strings involves finding an alignment by considering 3 modes of matching; these are exact matching, stemmed matching, and synonym matching, selecting whichever returns the largest score. Unigram precision P is calculated much as in BLEU, but based on the mappings. Then, unigram recall R is also calculated. These two metrics are combined and, to balance out the benefit to longer strings, a penalty is calculated based on the number C of mapped unigrams which are aligned positionally between the two strings. In METEOR, unlike in BLEU, if multiple references are provided, the best score is selected.

$$METEOR = \frac{10PR}{R + 9P}(1 - 0.5\frac{C}{\#maps})^3$$

ROUGE Recall-Oriented Understudy for Gisting Evaluation (ROUGE) was also developed in the wake of BLEU in Lin (2004). It has an array of versions including n-gram matching, longest common subsequence, and skip bigram co-occurrence. We focused only on n-gram matching, specifically unigrams. This form calculates precision the same way BLEU does. As we had multiple references, we returned the average score for each image. It is worth noting that unlike BLEU and METEOR, ROUGE does not impose a penalty. The implementation we used calculated both precision and recall, with the former essentially giving an unpenalized BLEU-1 score.

We also considered two other metrics, Consensus-based Image Description Evaluation (CIDEr by Vedantam et al. (2015)) and Semantic Propositional Image Caption Evaluation (SPICE by Anderson et al. (2016)) but did not employ them for this project. CIDEr works using TF-IDF weighted (1-4)-grams and cosine similarity between candidates and reference; the TF-IDF stage requires an entire corpus. SPICE relates strings by syntactic parsing of the strings using a pre-trained dependency parser, then converting those into 'scene graphs' and comparing those scene graphs; this was judged as prohibitively involved for this project.

4. Results

4.1 Quantitative Results

Our quantitative results can be seen in Figure 3. We calculated BLEU-2, BLEU-3, METEOR, and ROUGE-1 scores for all three decoder models (Papineni et al. (2002); Banerjee and Lavie (2005); Lin (2004)). The most immediate observation is that the RNN model under-performed compared to the LSTM and Transformer models. This is evident by the BLEU scores and the ROUGE precision scores, though the RNN did better on METEOR and ROUGE recall scores than was expected. For reasons discussed in the next section, the RNN gave much longer predictions than the other two models. These longer predictions contributed to higher recall scores (METEOR is based in part on recall) because the model's predictions were not punished for longer sequences. The RNN was likely finding a good amount of unigram matches, but maybe only by virtue of its longer predictions.

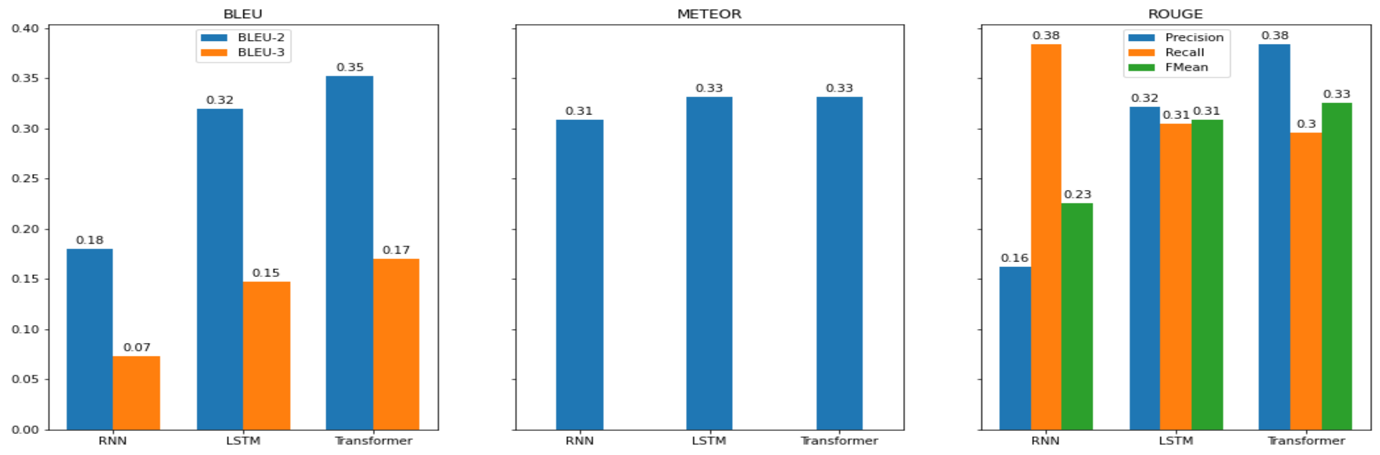


Figure 3: Left: BLEU-2 and BLEU-3 scores. Canter: METEOR scores. Right: ROUGE1 precision, recall, and fmean scores.

The LSTM and Transformer models, on the other hand, performed much better on precision scores, while still slightly outperforming the RNN on METEOR (because METEOR is also precision based). These two models clearly did a better job of finding all n-grams. The Transformer model takes the lead in terms of precision scores, while more or less matching the LSTM for recall. The Transformer is especially adept at finding matching unigrams compared to the LSTM model, given the ROUGE precision scores, while the LSTM has an incrementally better ROUGE recall score.

These last results are more significant than they seem when we consider the training time. The RNN and LSTM both took about 10 epochs before training loss converged, requiring about 10-20 hours on our machines. However, the Transformer model took significantly less time, finding a mostly stable loss within just a couple epochs and finishing after no more than 7 epochs, requiring 3-5 hours.

4.2 Qualitative Results

Qualitatively, there is a stark difference between the RNN captions and the LSTM and Transformer captions. What is immediately clear is that the RNN decoder never learned to structure or stop its sentence, even if it often found one or two relevant words. Instead, it has a tendency to end up in loops until it runs out of space, as can be seen in Appendix A. The LSTM and Transformer were much more successful in generating reasonable captions. There were instances where either failed to generate a good caption, but even in many of those cases it is clear that the model had some idea of what was relevant in the image. For instance, the LSTM failed to identify a computer mouse, but still identified it as a blocky piece of technology.

It is difficult to compare the LSTM captions with the Transformer captions as both give consistently acceptable captions. The only noticeable disparities across the set are that the Transformer gave slightly more concise captions, which may have contributed to its better precision scores, and that the LSTM is slightly likelier to miss a key detail as in the cake example. Overall, all the models did a fair job of identifying the important objects in the image and associating the correct nouns, and the two more complex models were quite good at fitting those together.

5. Conclusion

We have assembled and tested three different decoders for image captioning; this allowed us to explore the natural language processing aspect of that challenge, as well as to explore the machine translation metrics used to evaluate the decoders' output. We found that two of our models developed a strong ability to describe the contents of images, though our simplest model struggled with the language portion of the task. Furthermore, we demonstrated that the Transformer decoder can match or exceed the LSTM decoder, while being much faster to train.

Time permitting, there are a few things we would do to make enhance this project. We would experiment with more encoder models, such as VGG19 or different configurations of EfficientNet. We would also test more variations of the Transformer, perhaps with more attention heads, to see how much better it could be made. Finally, we would use more of the metrics available both from the methods we used, e.g. ROUGE-2 and BLEU-1, and those we didn't, such as CIDEr and SPICE, to get a better sense of what the models are doing well or poorly and how the metrics compare to each other.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation, 2016. URL `https://arxiv.org/abs/1607.08822`.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare R. Voss, editors, *IEEvaluation@ACL*, pages 65–72. Association for Computational Linguistics, 2005. URL `http://dblp.uni-trier.de/db/conf/acl/ieevaluation2005.html#BanerjeeL05`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Keras.io. Keras documentation: Image captioning. URL `https://keras.io/examples/vision/image_captioning/#model-training`.

Chin-Yew Lin. Rouge: a package for automatic evaluation of summaries. 2004.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. URL `https://arxiv.org/abs/1405.0312`.

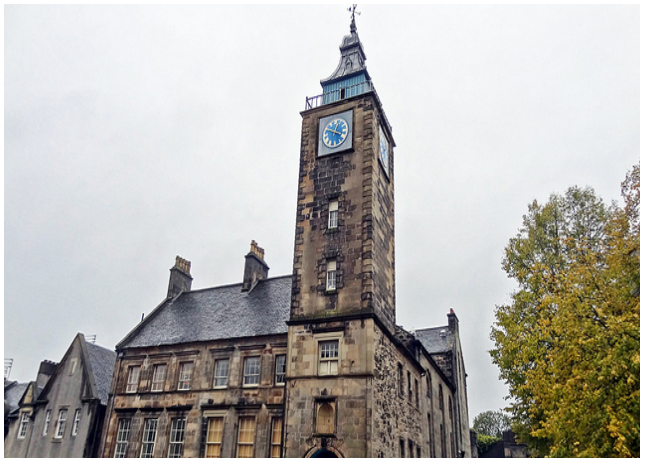
Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019. doi: 10.48550/ARXIV.1905.11946. URL `https://arxiv.org/abs/1905.11946`.

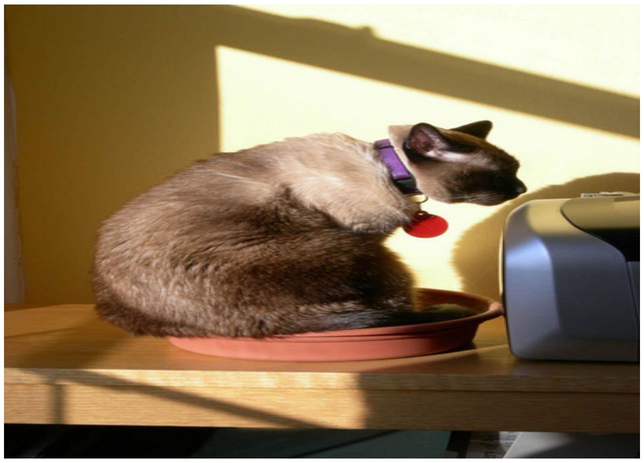
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL `https://arxiv.org/abs/1706.03762`.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575, 2015. doi: 10.1109/CVPR.2015.7299087.

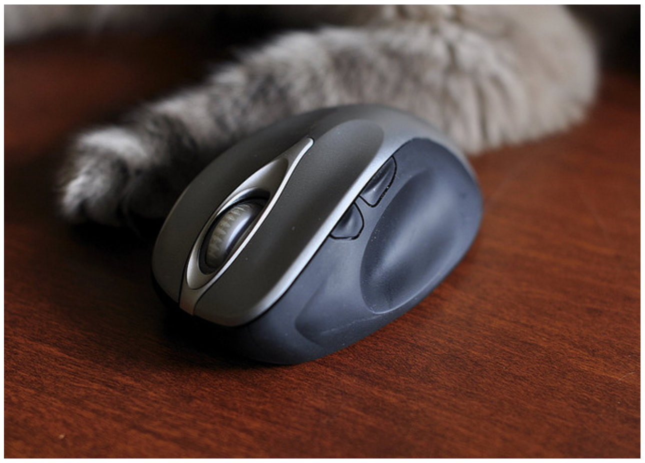
Appendix A.



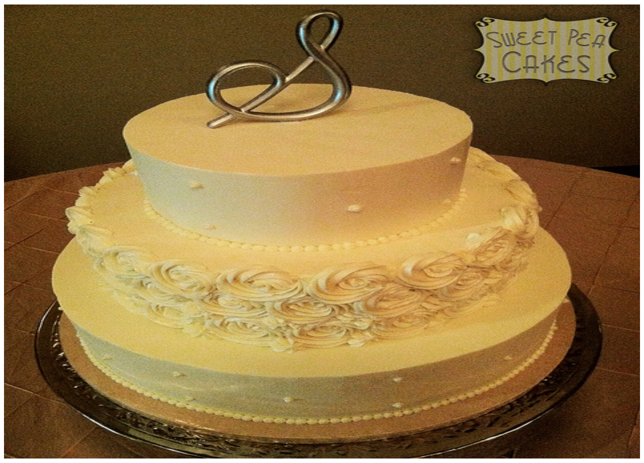
REF: A very large clock tower on the side of a church.
An old building with a clock at the top of it.
The tall clock tower is built into the corner edge of the building.
An old building has a clock tower with a weather vane.
An old building's clock tower is being displayed.
RNN: ben clock tower with a clock on it and a clock tower in the background of a building with a clock tower in the background
LSTM: a large building with a clock tower in the middle of it
T: a clock tower is shown in the middle of the street



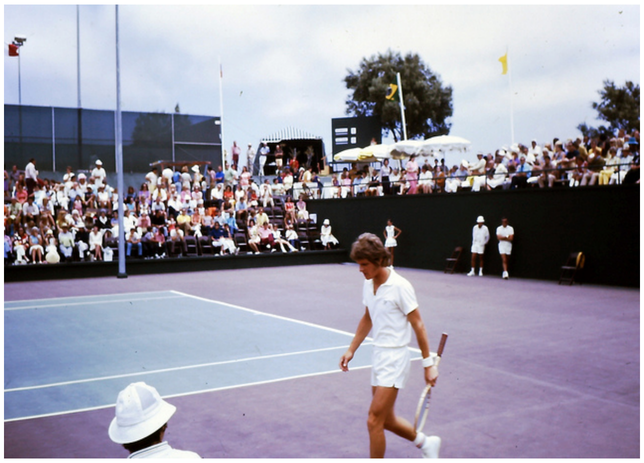
REF: A cat sitting in a bowl on a table.
A cat is sitting inside of a dish.
A cat with a purple and red collar sitting in a bowl
a brown and white Siamese cat is sitting on a table
A siamese cat sitting on top of a table.
RNN: cat laying on a white bed with a white cat sitting on top of a table with a laptop on it and a keyboard on
LSTM: a cat is sitting on a table with a spoon
T: a cat is lying on a plate on a table



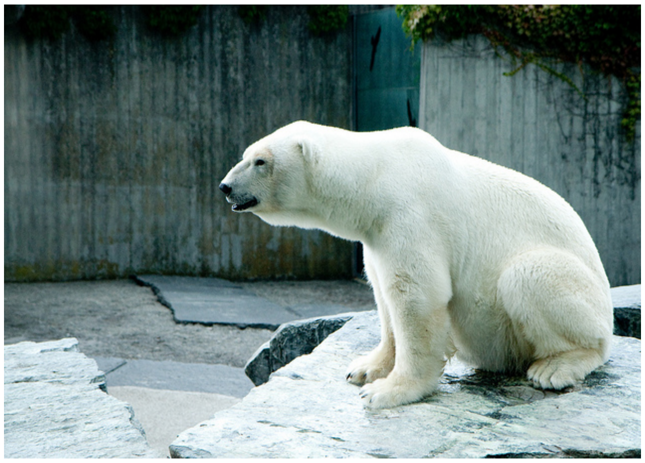
REF: a wireless computer mouse on a wooden table
A cat tail next to a mouse
A mouse kept just besides a cat's tail.
A cat with it's paw next to a computer mouse on a wooden table.
A computer mouse with a cat's paw next to it.
RNN: deserts on a plate on a table with a knife on it and a cup of coffee on it and a cup of coffee on
LSTM: a person holding a cell phone in their hand
T: a close up of a mouse on a table



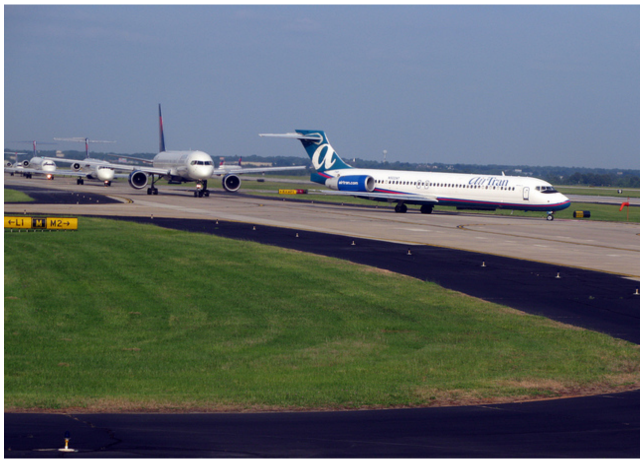
REF: A white wedding cake with a decorative top.
A three layer white cake with a symbol on top.
A three tiered wedding cake in white is shown with a large S on top.
An elegant wedding cake topped with a cursive S.
This beautiful wedding cake has a cursive s on top
RNN: white cake with a white and white checkered cloth with a white top and a white plate on it and a cup of coffee on
LSTM: a yellow and yellow fire hydrant sitting on a sidewalk
T: a cake with a red and yellow and green and yellow



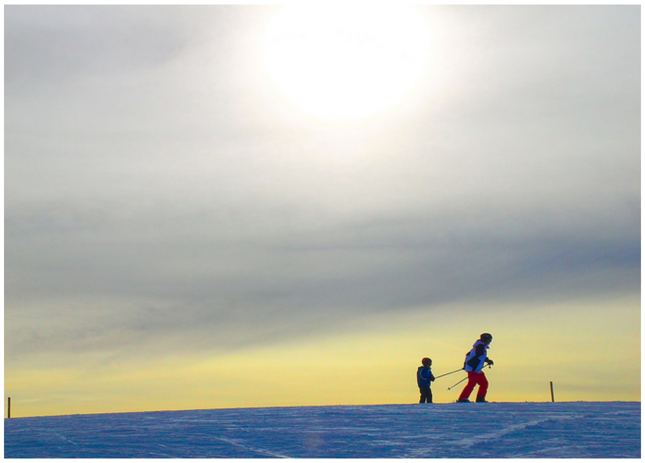
REF: A man with a tennis racket plays a game.
A crowd gathered to watch a tennis match.
Man walking onto the court during a professional tennis match
An old picture of a person playing in a tennis game.
A tennis player walking on the court during a match
RNN: outfielder in a baseball game of a match of fans watching him to serve the ball in the air on a tennis court with a
LSTM: a man is playing tennis on a tennis court
T: a tennis player is standing on a tennis court



REF: A polar bear sitting on some rocks.
A large white polar bear sitting on top of a rocky ground.
A polar bear sitting on a stone in his exhibit.
A polar bear sits in the sun and dries off.
A polar bear sitting on some ice by a fence.
RNN: polar bear standing by a pool of water with its paws on its paws on a persons legs of its paws on a shoe on
LSTM: a polar bear is laying down on a rock
T: a polar bear is sitting on a rock



REF: A group of airplanes that are sitting on a runway.
some white jets are lined up on a runway
Several airliners taxiing one by one on the tarmac
A line of airplanes waits to take off at an airport.
Modern jet airplanes lined up on the runway ready for take off
RNN: airways express plane on a runway at an airport terminal at an airport runway at an airport terminal at an airport terminal at an airport
LSTM: a large passenger jet sitting on top of a runway
T: a large passenger jet on the runway of an airport area



REF: A photo of two people skiing on the snow.
A man skiing on snow besides a child.
A picture of a couple people skiing in the snow.
Two people in the distance skiing against the horizon.
Two skiers in the distance under a cloudy sky.
RNN: people are flying kites on a beach near the ocean with a man on the beach and a kite in the background and a man
LSTM: a man is flying a kite in the sky
T: a person on a snowboard is jumping over the ocean