9-2020

Tag: Automated Image Captioning

Nathan Funckes
Grand Valley State University

Tag: Autonomous Image Caption Generator

Nathan Funckes
School of Computing
Grand Valley State University
funckesn@mail.gvsu.edu

Erin Carrier
Advisor
carrieer@gvsu.edu

Greg Wolffe
Advisor
wolffe@gvsu.edu

Abstract

Many websites remain non-ADA compliant, containing images which lack accompanying textual descriptions. This leaves sight-impaired individuals unable to fully enjoy the rich wonders of the web. To address this inequity, our research aims to create an autonomous system capable of generating semantically accurate descriptions of images. This problem involves two tasks: recognizing an image and linguistically describing it. Our solution uses state-of-the-art deep learning: employing a convolutional neural network that "learns" to understand images and extracts their salient features, and a recurrent neural network that learns to generate structured, coherent sentences. These two networks are merged to create a single model that takes as input arbitrary images and outputs relevant captions. The model's accuracy is quantified using various language metrics, such as the Bilingual Evaluation Understudy designed to rate language translation systems. After training, we hope to validate our approach by deploying our model on local, online social media feeds.

1. Introduction

With the rise of the Web and the expansion of social media, online images are ingrained into our daily lives. Unfortunately, many of these images lack accompanying descriptions. This leaves the world's sight-impaired population, including the over 1 million legally blind Americans [55], unable to fully enjoy the rich wonders the online world has to offer. In order to address this inequity, our research aims to use deep learning techniques to create an autonomous system capable of generating semantically accurate descriptions of untagged images, which can then be read and vocalized by assistive technology.

Deep learning [27] is a domain of computer science that is a subfield of machine learning, which is itself a subfield of artificial intelligence. Deep learning is focused on the use of artificial neural networks, computer algorithms based loosely off of how the brain works. These artificial neural networks are made up of layers of nodes connected by weights as seen in Figure 1. These networks function by taking in a vectorized input, which is transformed by the weights generating the values for the next layer. This process is repeated through the layers of the network, allowing for multiple levels of abstraction, until the output value is generated. When the network is training, the generated output is compared to the provided ground truth output, allowing the model to update its weights and "learn" through the use of backpropagation [45].

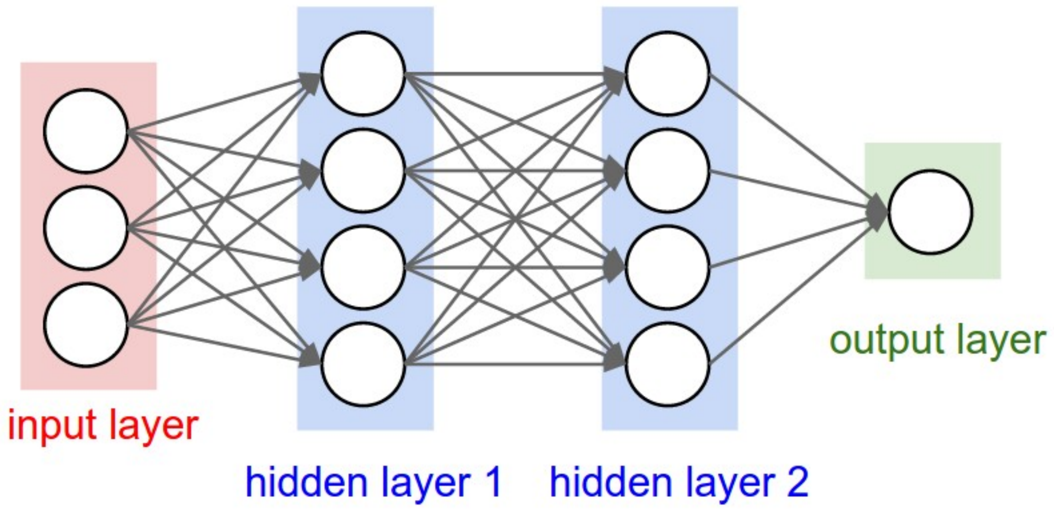


Figure 1: A traditional feed forward neural network. Image from [20]

Our approach to the problem of image captioning combines two specialized deep neural network architectures: a convolutional neural network architecture is used to extract the features of the image, and a recurrent neural network architecture is used to generate the description of the image.

2. Background

Automated image captioning is a problem that stands at the intersection of two major areas of study in deep learning, computer vision (CV) and natural language processing (NLP). To help explain their use in CV and NLP, an overview of convolutional neural networks is provided in section 2.1 and an overview of recurrent neural networks is given in section 2.2.

2.1. Convolutional Neural Networks

Convolutional neural networks (CNNs or ConvNets) were first introduced in 1998 by LeCun *et al*. [28] with the LeNet Architecture. This foundational CNN was later significantly improved by Krizhevsky *et al*. [25] with the introduction of a new architecture dubbed AlexNET. The improvements to the CNN architecture combined with the use of GPUs (Graphics Processing Units) allowed AlexNet to win the ImageNet [46] competition in 2012 with a significant improvement over other models. This breakthrough ignited an explosion of interest in the field of machine learning and provided a foundational architecture that has been continually improved upon [12, 49, 53]. Convolutional neural networks' success in computer vision is evident not only through the ImageNet competition, but can also be seen in real-world applications such as medical imaging [33] and autonomous (self-driving) vehicles [62]. Figure 2 provides an overview of how CNNs accomplish such tasks.

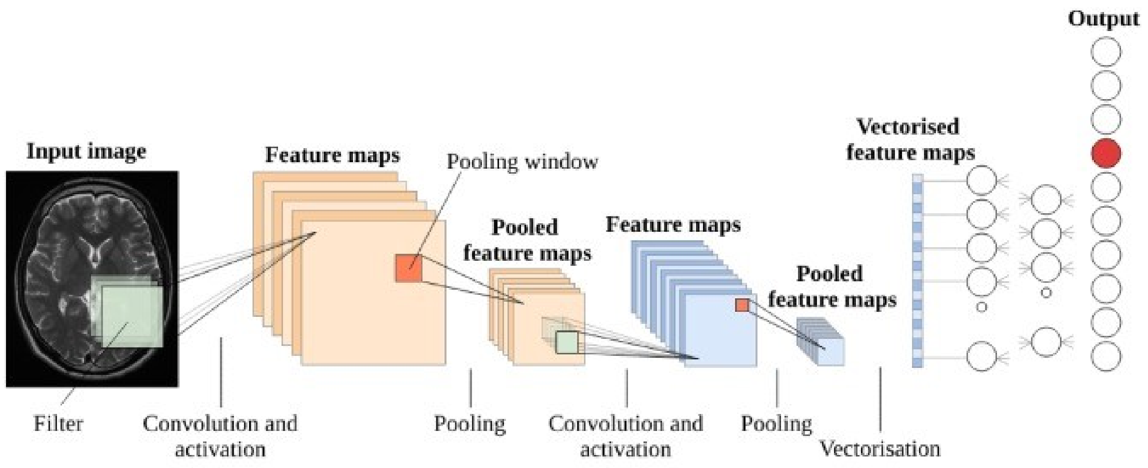
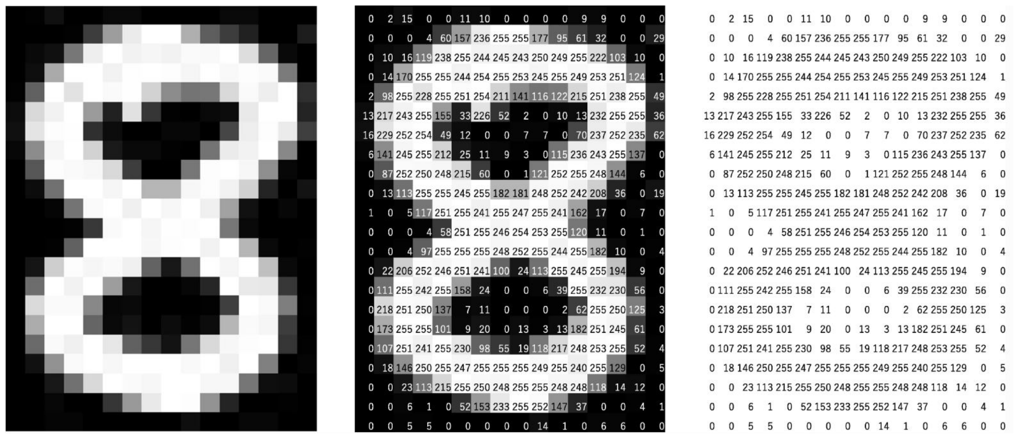


Figure 2: An overview of how a basic CNN architecture works. Image from [33]

In computer vision problems convolutional neural networks work by taking in a vectorized image, where an image is represented by its pixel values in a grid structure, as seen in Figure 3. These vectors go through layered operations that allow convolutional neural networks to extract features and form high-level representations of images. These layers consist of convolutional layers (described in section 2.1.1) and pooling layers (discussed in section 2.1.2).



0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

Figure 3: Computers see images as arrays of numbers with values ranging from 0 to 255, representing each pixel's brightness. Image from [63]

2.1.1 Convolutional Layer

The convolution operation, depicted in Figure 4, is a linear operation responsible for feature extraction in CNNs. In the convolution operation a small matrix of numbers, called a filter, steps through the input array. At each position of the input the element-wise product between the elements of the filter and the input array is computed and the values are summed. This sum is then placed in the corresponding location of the output array, called a feature map. In a convolutional layer multiple filters are applied to form an arbitrary number of feature maps determined by the hyper-parameter for the number of filters. The filters are the weights a CNN learns during training.

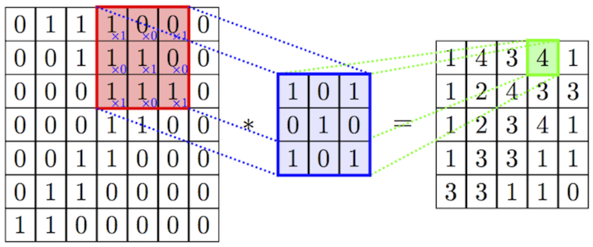


Figure 4: An example of the convolution operation computing a feature map. Image from [37]

The convolution operation described above results in a feature map with a smaller size than the input array. In order to prevent this and conserve the size of the array, padding can be applied to the input array. Padding, more specifically zero padding, is the technique of adding additional rows and columns of zeros to the sides of the input array so that the resulting feature map retains the size of the input as seen in Figure 5.

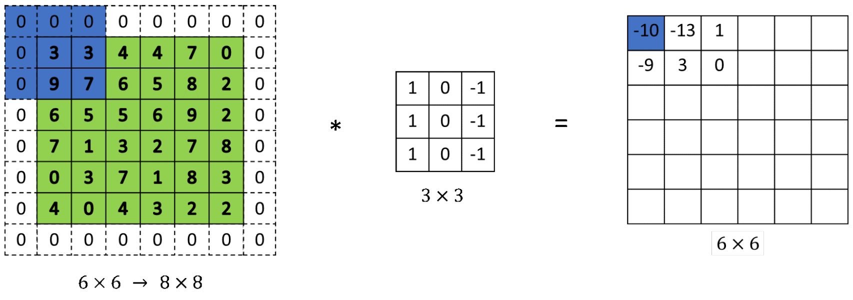


Figure 5: An example of zero padding being applied to maintain the size of the array. Image from [11]

An important aspect of the convolutional layer is the implementation of weight sharing. This ability allows the learned filters to be applied to every position of the image. This means features only need to be learned once and can then be found across all image positions, increasing the model's efficiency and reducing the number of parameters required to learn and adequately extract the features of an image.

The convolution operation is a linear operation; so, in order to allow for the representation of non-linear relationships, its outputs are passed through a non-linear activation function such as the Rectified Linear Unit ($ReLU$) function [38].

2.1.2 Pooling Layer

The pooling operation is the process of down-sampling the feature map outputs of convolution layers. This down-sampling reduces the number of total parameters of the network, while simultaneously allowing the network to learn increasingly abstract image features. There are different types of pooling, the most commonly used is max-pooling. In the max-pooling operation, a small filter steps through the feature map saving only the largest of the values at each location to a new array of a smaller size. This is demonstrated in Figure 6.

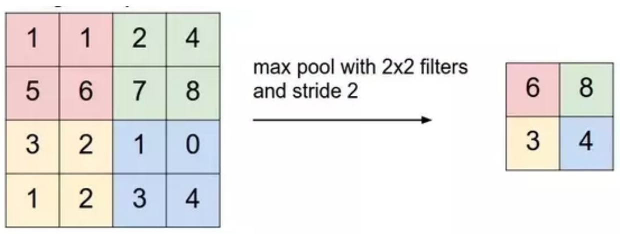


Figure 6: Max pooling being applied across a feature map. Image from [20]

2.2. Recurrent Neural Networks

Traditional neural networks lack the ability to recall previous information. Recurrent neural networks address this shortcoming by including feedback loops in their structure. This is visualized in Figure 7 in which A takes in the input x_t and outputs a value h_t and also contains a loop to pass information to the next timestep in the network. The use of a feedback loop in a neural network may be better visualized by unrolling the loop and thinking of it as multiple copies of the same network. It sends messages on to the next copy as seen in Figure 8, where A contains t steps. At each step, an input x is taken in, the value h is output, and select information is passed onto the next step.

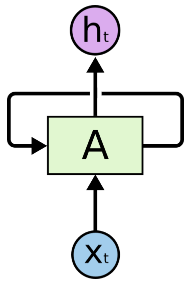


Figure 7: A basic RNN structure. Image from [39]

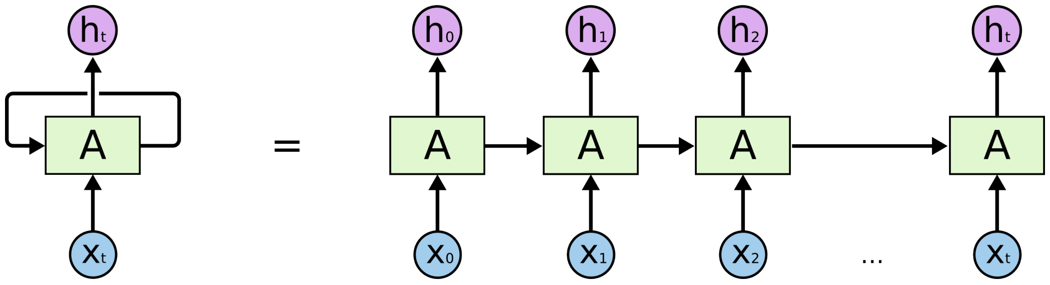


Figure 8: An RNN unrolled. Image from [39]

In theory, the inclusion of loops in RNNs means that they should be capable of learning how to solve problems that rely on long-term dependencies, but in practice RNNs struggle to learn these long-term dependencies[13, 5]. However, an advanced RNN called the Long-Short Term Memory (LSTM) architecture, seen in Figure 9, is designed to address this issue.

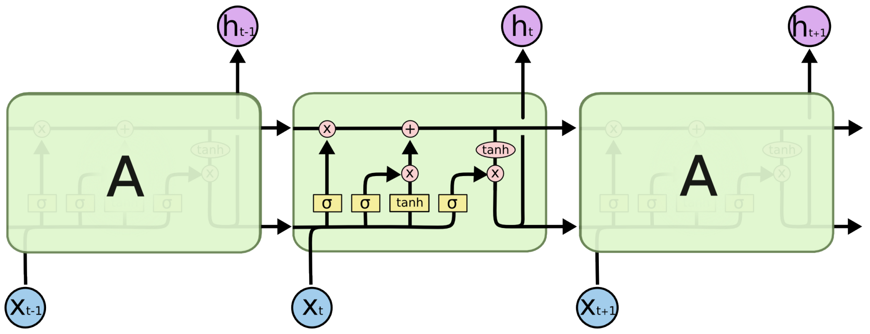


Figure 9: LSTM architecture. Image from [39]

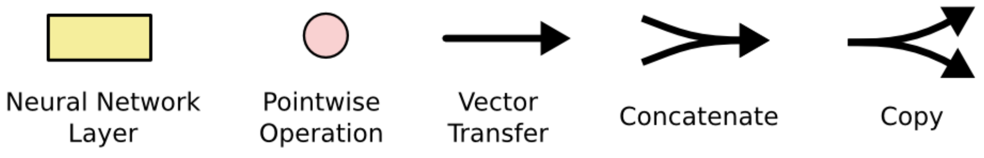
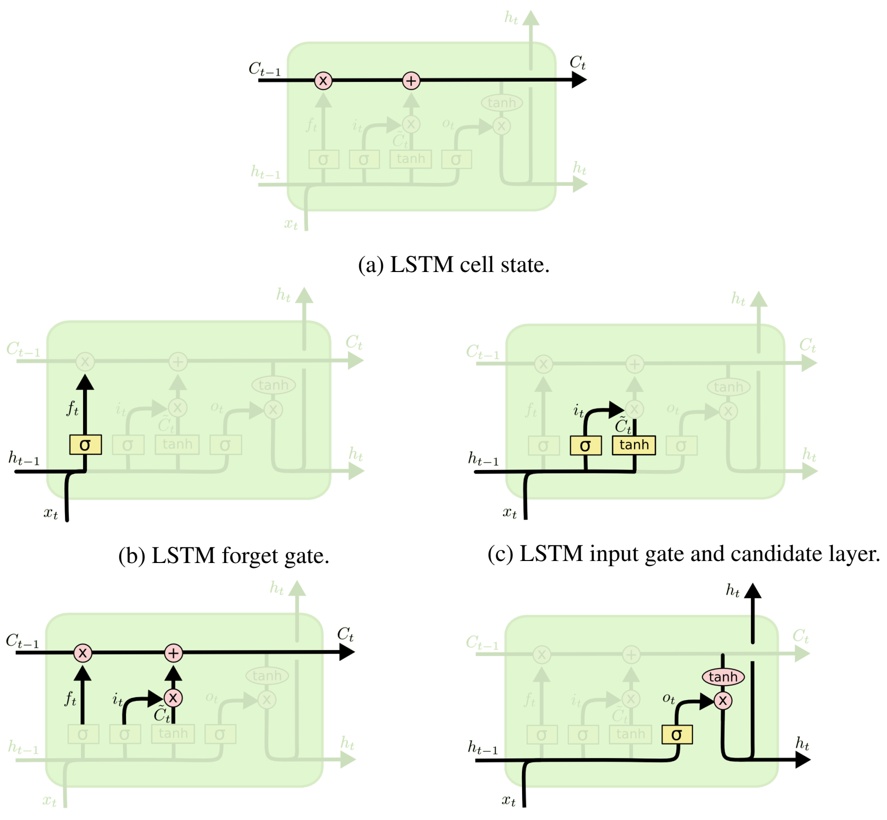


Figure 10: LSTM notation. Image from [39]

LSTMs are an advanced RNN introduced by Hochreiter *et al.* [14] and specifically designed to be able to learn long-term dependencies. The biggest difference between a typical RNN and an LSTM is that, while an RNN contains a single neural network layer, an LSTM contains four layers. The inner structure of the LSTM architecture can be seen in Figure 9 and its notation is detailed in Figure 10. Figure 11 displays the LSTM's inner-workings step by step similar to that given in [39].



(a) LSTM cell state.

(b) LSTM forget gate.

(c) LSTM input gate and candidate layer.

(d) LSTM cell state update.

(e) LSTM output gate.

Figure 11: Visualization of LSTM gates. Images from [39]

The top horizontal line highlighted in Figure 11a is the LSTM's cell state. The cell state can be thought of as the memory of the LSTM, containing previously seen information. The LSTM network layers, called gates, control what information is sent to the cell state, determining the information that is "remembered". These gates are made up of a sigmoid neural net layer and a point-wise operation. The sigmoid function outputs values 0 to 1, where in this case a 0 results in no information passing through the gate and a 1 results in everything passing through the gate.

The forget gate layer highlighted in Figure 11b is the first step of the LSTM. The forget gate determines what information from the previous cell state is kept and what information is discarded. The forget gate, represented as $f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$, takes in h_t and x_t and outputs a vector of values between 0 and 1. If the output is 0 then the entire previous state is "forgotten", if the output is 1 then the entire previous state is "remembered".

The next step is the input gate and $\tanh$ layer, highlighted in Figure 11c. This step determines what new information is going to be stored in the cell state. This is a two-part process. First, the input gate, represented by $i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$ determines which values will be updated. This is accomplished by passing the h_{t-1} and x_t vectors through the sigmoid layer. Next, the $\tanh$ layer, represented by $\tilde{C}_t = \tanh(W_C * [h_{t-1}, x_t] + b_C)$, computes a potential new cell state.

The next step, highlighted in Figure 11d, uses the outputs of the previous two steps to update the old cell state C_{t-1} to the new cell state C_t. This is a three part process represented by $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$. First, the previous cell state C_{t-1} is multiplied by the forget vector f_t, allowing the old cell state to forget the information selected by the forget gate. Next, the vector containing the values chosen to be updated by the input gate i_t and the vector containing the potential new cell state C_t are multiplied together to provide the new cell values. These values are then added to the previous cell state creating the new cell state which will be passed to the next cell.

Finally, the output must be determined; this is the two-part process highlighted in Figure 11e. First the output gate, represented by $o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$, determines what information from the inputs h_{t-1} and x_t is relevant to the output. The next step, represented by $h_t = o_t * \tanh(C_t)$, multiplies the vector o_t and a filtered version of the cell state, normalized between -1 and 1 by the $\tanh$ function. This final vector h_t is then output.

3. Foundations of Image Captioning

Image captioning is a long-standing problem in the field of artificial intelligence and computer vision. Early approaches to image captioning usually employed either template-based methods [65, 29, 36, 54] or retrieval-based methods [10, 15, 40, 51]. Both template-based methods and retrieval-based methods utilize hard-coded rules and hand-engineered features, putting obvious limitations on the captions generated. More recently, researchers have made substantial progress by applying deep learning to image captioning. What follows is an overview; a deeper look into the various approaches and recent advances in image captioning can be found in [6, 26, 3, 17, 50].

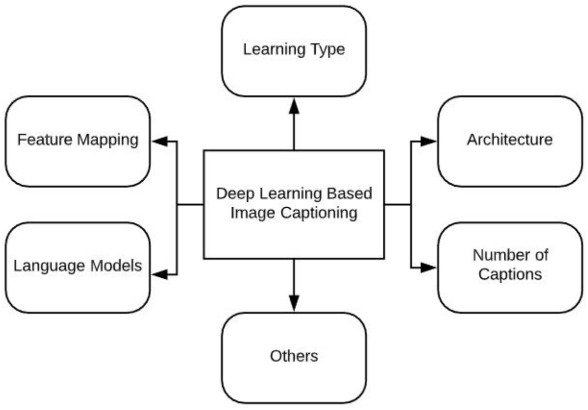


Figure 12: Types of image captioning in deep learning.

When approaching image captioning with deep learning there are numerous choices and decisions that must be considered. The most prominent decisions required in deep learning approaches can be grouped into the six categories seen in Figure 12.

3.1. Feature Mapping

Deep learning approaches can use either visual space or multimodal space in order to generate image captions. Both approaches train on datasets containing images with corresponding captions. In the visual space approaches the features of the image are extracted and sent to the language encoder independent of the caption. However, in a multimodal approach, a multimodal space is learned from the image and caption and then this multimodal space is sent to the language-encoder. Although there are examples of multimodal approaches [22, 21, 23, 35], deep learning using a visual space approach is the most common method, and is discussed further in the remainder of this section.

3.2. Learning Type

The majority of deep learning approaches to image captioning use supervised learning, training on images with paired captions, but there have been a few instances where researchers have used other learning types: unsupervised learning

(learning from unlabeled data) [48, 8], and reinforcement learning (learning based on actions and rewards) [24, 42, 43].

3.3. Architecture

The two main architectures used in image captioning are the encoder-decoder architecture as used in automated translation [52], and the compositional architecture.

In a typical encoder-decoder approach, a CNN is used to extract the different features of the image. These features are then passed to the language model, which uses the image features to generate words that are combined to make a caption. In a typical compositional approach, a CNN is used to extract image features, these features are then passed to the language model which generates multiple potential captions. These captions are then ranked using a deep multimodal similarity model, and the highest quality caption is chosen. Examples of the encoder-decoder architecture can be found in [59, 60, 34], and examples of the compositional architecture are presented in [9, 57, 58].

3.4. Number of Captions

In image captioning, another decision to make is whether to generate captions for the whole image or generate separate captions for regions of the image [18, 64]. Generating captions for the whole image is more popular, due at least in part to the fact that most datasets only contain captions for the whole image.

3.5. Others

In addition to the above approaches, there are also a number of methods that do not quite fit into the above groupings. These include attention-based, semantic concept-based, novel object-based methods, and stylized captions.

3.6. Language Models

Methods can also be differentiated based on the language model they use. LSTMs are currently the most popular language model used in image captioning, but recently there have been a few approaches that use language-CNNs instead of an RNN or LSTM [61, 2].

3.7. Datasets

There have been a number of different datasets published that present image caption pairs for the purposes of investigating and solving the problem of image captioning. The most prominent datasets are Flickr8k [16], containing a total of 8,000 images, Flickr30k [66], containing 30,000 images, and MSCOCO [32], containing more than 300,000 images. Each of these datasets contains images with five unique captions per image. This research project used Google's conceptual captions dataset [47] containing 3.3 million images, each with only one unique caption.

4. Preparing the Data

Google's dataset provides the ground truth captions and links to the images online. Using this dataset first required downloading the numerous images. However, since these images are scraped from the web, some of the links were no longer valid and some of the images were corrupted. This resulted in a smaller dataset than originally posted, as well as the requirement for extensive data processing as the broken links had to be ignored and the corrupted images deleted. It also required the deletion of the captions related to these "bad" images. Overall, this process reduced the training set to about 87% of the original data (from 3,318,333 to 2,902,667 image-caption pairs), and the validation set to approximately 85% of the original data (from 15,840 to 13,501 image-caption pairs). However, initial experiments in this project used only 150,000 of the remaining training images while still validating on all 13,501 data points in the validation set. This decision was made to balance the benefits of a large dataset with the long training times that are not conducive to experimentation.

Preparing the images and captions for training involved extracting the image features and cleaning the captions. The image feature extraction component utilized VGG-16. In order to reduce training time, the images were processed through VGG-16 independently from the rest of the model, saving the image features to a file and using them as an input later in the workflow pipeline. The captions were cleaned by removing any punctuation prior to training . Additionally, a tokenizer was created for the captions. A tokenizer maps each unique word to an integer. This allows text to be converted to numbers and used as an input to the model as well as generated words for the model's output.

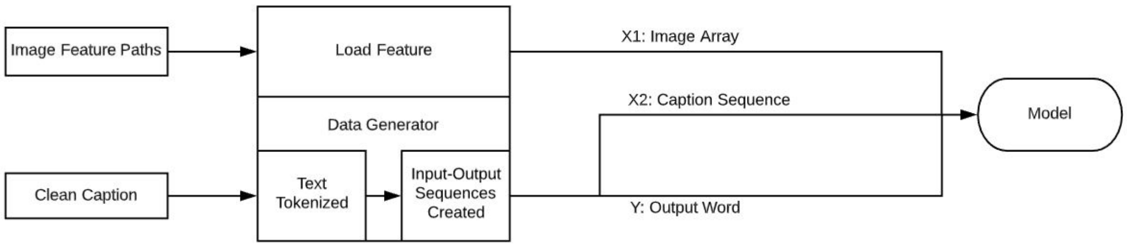


Figure 13: An overview of the data generator.

Training of the model involved implementation of a data generator, which allowed for training using large amounts of data without overloading the memory of the machine. The data generator method supports this by only preparing the necessary data for one batch at a time. This process, seen in Figure 13, involved loading in the saved image feature and creating the tokenized caption input and output sequences. The data generator prepares this data for each image-caption pair in each batch and then sends it to the model, and the process is repeated for each batch.

5. Approach

5.1. Model

The approach taken in this research involves a merge model, implemented as an encoder-decoder architecture. The model uses the encoded image features, extracted by a CNN, as an input as well as the encoded previously generated sequence which is passed through an LSTM. The output of the image features and LSTM are combined and input to a feed-forward neural network, which uses them to predict the next word in the sequence. Each layer in the network has weights that are "learned" throughout the training process.

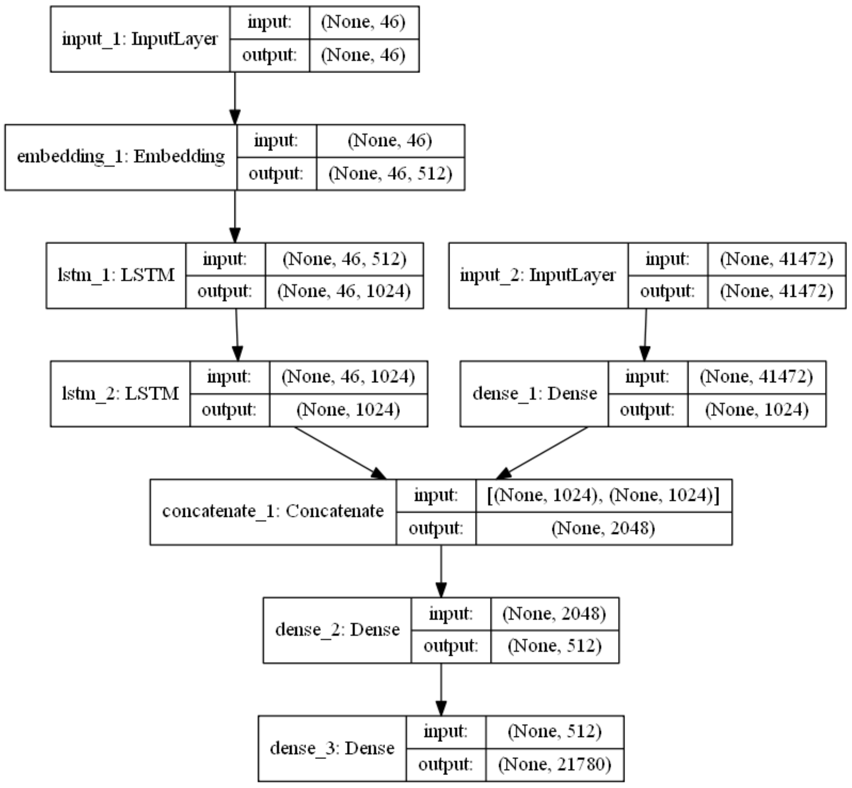

input_1: InputLayer | input: (None, 46) | output: (None, 46)
embedding_1: Embedding | input: (None, 46) | output: (None, 46, 512)
lstm_1: LSTM | input: (None, 46, 512) | output: (None, 46, 1024)
input_2: InputLayer | input: (None, 41472) | output: (None, 41472)
lstm_2: LSTM | input: (None, 46, 1024) | output: (None, 1024)
dense_1: Dense | input: (None, 41472) | output: (None, 1024)
concatenate_1: Concatenate | input: [(None, 1024), (None, 1024)] | output: (None, 2048)
dense_2: Dense | input: (None, 2048) | output: (None, 512)
dense_3: Dense | input: (None, 512) | output: (None, 21780)

Figure 14: Network architecture of the model.

The specific model architecture is illustrated in Figure 14. The image features extracted by VGG-16 are used as one of the inputs and the tokenized form sequence of the previous words in the caption comprise the other input.

The tokenized word sequence is taken into the model as `input_1` and then passed to an embedding layer. The embedding layer takes the tokenized sequence and outputs a vector representation of the sequence of words. This output is passed to the first LSTM layer, whose output is then passed onto the next LSTM. These LSTM layers produce a vector representation of the next word probabilities based on the prior words in the caption.

The image features previously extracted by VGG-16 are taken in as the second input of the model. This vector of image features is passed to a dense layer which reduces the dimensionality of the features and outputs a higher-level representation of the image features.

These two outputs, the next word probabilities and the higher-level image features, are combined and passed to a dense layer, which outputs a vector representation of the relationship between image features and potential next words. This output is passed to the final dense layer which outputs the predicted next word in the sequence.

5.2. Evaluation Metrics

5.2.1 BLEU

BLEU (Bilingual Evaluation Understudy) [41] is an automated metric used to measure the quality of generated text. It does have some limitations: BLEU is only a good evaluation metric for short text [7] and there are some cases where a better BLEU score does not necessarily correspond to higher quality text [31].

1-gram	'I', 'love', 'to', 'code'
2-gram	'I love', 'love to', 'to code'
3-gram	'I love to', 'love to code'
4-gram	'I love to code'

Table 1: n-grams

The BLEU metric uses n-grams, ordered sets of words (see Table 1), to evaluate generated or candidate text through a modified version of the precision measure. The standard precision measure is computed by adding up the number of words or unigrams in the candidate text that also appear in the reference text. That count is then divided by the number of unigrams in the candidate text, providing a ratio of words in reference to total words. The basic precision measure runs into a problem when the candidate text has an overabundance of reference unigrams, which causes high precision scores but poor quality text. However, this problem can be combated by considering a reference word to be exhausted after the first matching candidate word is found, a technique called clipping the count. Therefore BLEU uses a modified version of the precision measure: modified n-gram precision.

Modified n-gram precision is computed similarly for each n. For each value of n the n-grams in the candidate text are counted and the maximum n-grams in the reference are counted. The candidate counts are then clipped by the maximum reference counts, summed, and divided by the total number of n-grams in the generated text. This is represented by Equation 1.

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{–}gram \in C} Count_{clip}(n\text{–}gram)}{\sum_{C' \in \{Candidates\}} \sum_{n\text{–}gram' \in C'} Count(n\text{–}gram')} \tag{1}$$

In addition to the modified precision formula, the BLEU metric uses a brevity penalty which penalizes generated sentences for being shorter than the reference sentence. The brevity penalty does not affect sentences longer than the reference sentence as they are already penalized by the modified precision. This brevity penalty is computed according to Equation 2, where c represents the candidate text length and r represents the reference length.

$$BP = \begin{cases} 1 & if\ c > r \\ e^{(1-r/c)} & if\ c \leq r \end{cases} \tag{2}$$

The overall formula for BLEU is given in Equation 3, where N represents the maximum n-gram size used and the positive weights w_n sum to one.

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} w_n \log p_n) \tag{3}$$

5.2.2 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [30] is another set of metrics that can be used to evaluate the quality of generated text. The ROUGE metrics compare word sequences, word pairs, and n-grams in the generated text to those in the reference text set. The different variations of ROUGE are used for different types of translation and generation tasks.

5.2.3 METEOR

METEOR (Metric for Evaluation of Translation with Explicit Ordering) [4] is a metric used to evaluate machine-translated text. METEOR is based on explicit word-to-word matches between the generated text and the reference text. Sentence stems and word synonyms are also considered by the METEOR metric to make it more versatile.

5.2.4 CIDEr

CIDEr (Consensus-based Image Description Evaluation) [56] is an automated metric designed specifically to be accurate at evaluating generated descriptions of images. CIDEr uses term frequency-inverse document frequency (TF-IDF) [44] in order to determine the human consensus of the generated text.

5.2.5 SPICE

SPICE (Semantic Propositional Image Caption Evaluation) [1] is another metric designed specifically to tackle the evaluation of generated image descriptions. SPICE is based on Scene Graph [19], a graph-based semantic representation. This method allows SPICE to extract information about objects, their attributes, and their relationships from the image descriptions and use this to more accurately evaluate the generated image description.

6. Results

6.1. Training Examples

One way to get a feel for a model's performance and whether or not it is learning is to examine a few example training images, observing how their captions evolved during the training process. The training process can be divided into runs, known as epochs, through the entire training set. Table 2 shows four different training images and their corresponding captions at different points throughout training. By investigating these examples we can see that, although the captions are not amazing, the quality seems to peak at epoch 30. In the examples we can see the captions improving as the model trains. This implies that our model is starting to learn how to accurately produce captions.

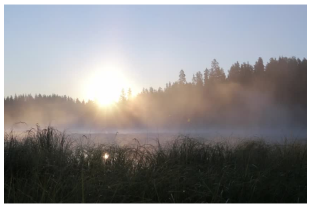
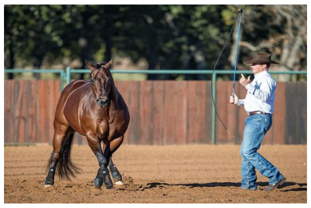
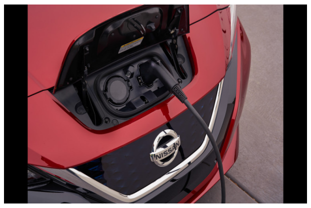
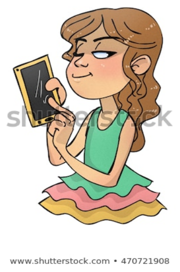
	Epoch	
	10	a man walks along a mountain towards the moon
	20	a city is a peninsula
	30	sunset over the ocean at the beach
	10	a young girl in a hat and hat is standing in the park
	20	a man with a bouquet of flowers in the park
	30	a young woman with a bouquet of flowers
	10	a man walks past a window of his car
	20	the car was erected in front
	30	a man walks to work on a petrol station
	10	illustration of a girl with a hula hoop
	20	illustration of a cute little boy holding a tray with joy
	30	vector illustration of a cute girl smiling

Table 2: Generated captions for a representative selection of training images at various points during the training process.

6.2. Evaluation Metrics on Training data

Looking at example caption predictions is a good way to subjectively evaluate a model's performance, but it is not scalable when processing large amounts of data. That task requires the evaluation metrics introduced earlier. These evaluation metrics are crucial for being able to test a machine learning model as it would take too much time for a human to manually rate thousands of caption predictions. Automated evaluation metrics allow for rapid and consistent scoring that can help guide research decisions while saving time and resources.

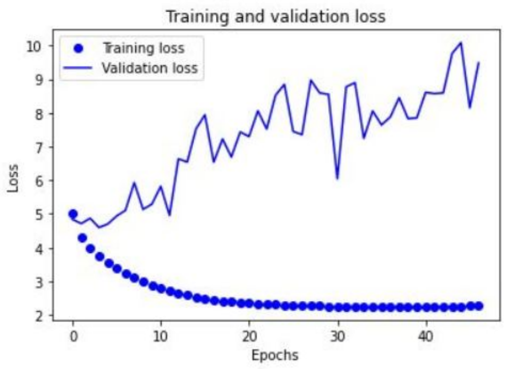
Epochs	10	20	30	40
BLEU-1	0.130252	0.137693	0.147615	0.143247
BLEU-2	0.068238	0.071931	0.079133	0.074483
BLEU-3	0.042996	0.045231	0.050497	0.047386
BLEU-4	0.030984	0.032895	0.035945	0.034264
ROUGE-L	0.162101	0.157243	0.163966	0.158511
CIDEr	0.287927	0.297938	0.343242	0.327422
METEOR	0.047937	0.048986	0.053577	0.050539
SPICE	0.042989	0.041677	0.045929	0.044928

Table 3: Evaluation metrics scores on 1000 training images at different points during training.

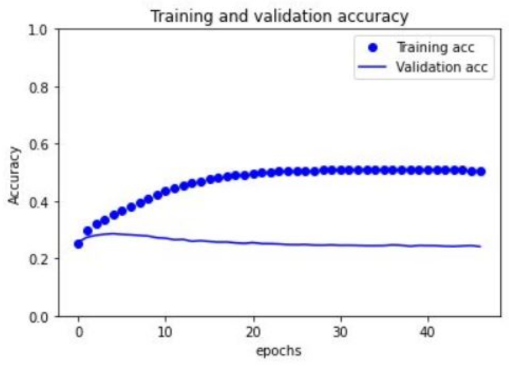
The scores given in Table 3 are the results of the various evaluation metrics computed over time, i.e. after the model has progressed through an increasing number of training epochs. The scores support the subjective observations: there is a general upwards trend for each score, peaking at epoch 30. This is helpful as it provides quantifiable evidence of improvement. However, the degradation in scores after epoch 30 indicates there is still work to do in improving the model.

6.3. Learning vs. Generalization

When approaching a problem using machine learning there are typically two main goals for a model. The first is for the model to be able to learn and solve the problem. The second is for the solution the model arrives at to be generalizable. Learning only focuses on how well the model is able to predict the right outputs on data it has seen before and is evaluated using the training dataset. Generalization is concerned with the model's ability to make correct predictions on new data and is typically evaluated using a validation or test dataset.



(a) Model's loss during the training period



(b) Model's accuracy during the training period

Figure 15: Graphs of model's performance on training and validation sets during the training process.

Two measurements for quantifying a model's progress (improvement occurring during training) are accuracy and loss. The accuracy of a model refers to how often the model predicts the right output for a given input. The loss of a model is similar but differs in an important way. The loss function does not look at how often the model is correct, but instead looks at the ”confidence” the model has in its prediction.

Loss and accuracy were measured throughout training and are visualized in Figure 15. It is important to note that in image captioning accuracy is not always the best indicator of how well a model is performing due to the complexity of language. This is the reason for using the automated evaluation metrics, which are capable of taking into account more information.

As seen in Figure 15 and supported by the earlier observations and scores the current model is capable of learning. Yet, it is also apparent that it struggles to generalize.

7. Conclusion and Future Work

The current model is able to generate captions of some quality after training on a subset of Google's conceptual captions dataset. Work remains in improving both the model's ability to learn and to generalize, requiring further experimentation with various design choices and a closer examination of the data. Due to the large size of the dataset, these further experiments will likely require significant amounts of time.

References

[1] P. Anderson, B. Fernando, M. Johnson, and S. Gould. SPICE: semantic propositional image caption evaluation. *CoRR*, abs/1607.08822, 2016.

[2] J. Aneja, A. Deshpande, and A. G. Schwing. Convolutional image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[3] S. Bai and S. An. A survey on automatic image caption generation. *Neurocomputing*, 311:291–304, 2018.

[4] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5:157–66, 02 1994.

[6] R. Bernardi, R. Çakici, D. Elliott, A. Erdem, E. Erdem, N. Ikizler-Cinbis, F. Keller, A. Muscat, and B. Plank. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *CoRR*, abs/1601.03896, 2016.

[7] C. Callison-Burch, M. Osborne, and P. Koehn. Re-evaluating the role of Bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy, Apr. 2006. Association for Computational Linguistics.

[8] B. Dai, D. Lin, R. Urtasun, and S. Fidler. Towards diverse and natural image descriptions via a conditional GAN. *CoRR*, abs/1703.06029, 2017.

[9] H. Fang, S. Gupta, F. N. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig. From captions to visual concepts and back. *CoRR*, abs/1411.4952, 2014.

[10] Y. Gong, L. Wang, M. Hodosh, J. Hockenmaier, and S. Lazebnik. Improving image-sentence embeddings using large weakly annotated photo collections. In *Computer Vision – ECCV 2014*, pages 529–545. Springer International Publishing, 2014.

[11] V. Gorade. Understanding cnns, Jun 2020.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[13] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[15] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 05 2013.

[16] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *J. Artif. Int. Res.*, 47(1):853–899, May 2013.

[17] M. Hossain, F. Sohel, and H. Laga. A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys*, 51(6):118, 2019.

[18] J. Johnson, A. Karpathy, and F. Li. Densecap: Fully convolutional localization networks for dense captioning. *CoRR*, abs/1511.07571, 2015.

[19] J. Johnson, R. Krishna, M. Stark, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678, 2015.

[20] A. Karpathy, J. Johnson, and L. Fei-Fei. Cs231n: Convolutional neural networks for visual recognition, 2016.

[21] A. Karpathy, A. Joulin, and F. Li. Deep fragment embeddings for bidirectional image sentence mapping. *CoRR*, abs/1406.5679, 2014.

[22] R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–595–II–603. JMLR.org, 2014.

[23] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539, 2014.

[24] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1008–1014. MIT Press, 2000.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[26] A. Kumar and S. Goel. A survey of evolution of image captioning techniques. *International Journal of Hybrid Intelligent Systems*, 14:1–19, 11 2017.

[27] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature Cell Biology*, 521(7553):436–444, May 2015.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] S. Li, G. Kulkarni, T. L. Berg, A. C. Berg, and Y. Choi. Composing simple image descriptions using web-scale n-grams. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 220–228, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[30] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[31] C.-Y. Lin and F. J. Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain, July 2004.

[32] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[33] A. S. Lundervold and A. Lundervold. An overview of deep learning in medical imaging focusing on MRI. *CoRR*, abs/1811.10052, 2018.

[34] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy. Generation and comprehension of unambiguous object descriptions. *CoRR*, abs/1511.02283, 2015.

[35] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. Explain images with multimodal recurrent neural networks. *CoRR*, abs/1410.1090, 2014.

[36] M. Mitchell, X. Han, J. Dodge, A. Mensch, A. Goyal, A. Berg, K. Yamaguchi, T. Berg, K. Stratos, and H. Daumé. Midge: Generating image descriptions from computer vision detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, page 747–756, USA, 2012. Association for Computational Linguistics.

[37] I. S. Mohamed. Detection and tracking of pallets using a laser rangefinder and machine learning techniques. *None*, 2017.

[38] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[39] C. Olah. https://colah.github.io/posts/2015-08-understanding-lstms/, 2015.

[40] V. Ordonez, G. Kulkarni, and T. L. Berg. Im2text: Describing images using 1 million captioned photographs. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 1143–1151, Red Hook, NY, USA, 2011. Curran Associates Inc.

[41] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[42] Z. Ren, X. Wang, N. Zhang, X. Lv, and L. Li. Deep reinforcement learning-based image captioning with embedding reward. *CoRR*, abs/1704.03899, 2017.

[43] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563, 2016.

[44] S. Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation - J DOC*, 60:503–520, 10 2004.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[47] P. Sharma, N. Ding, S. Goodman, and R. Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[48] R. Shetty, M. Rohrbach, L. A. Hendricks, M. Fritz, and B. Schiele. Speaking the same language: Matching machine to human captions by adversarial training. *CoRR*, abs/1703.10476, 2017.

[49] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[50] R. Staniute and D. Šešok. A systematic literature review on image captioning. *Applied Sciences*, 9:2024, 05 2019.

[51] C. Sun, C. Gan, and R. Nevatia. Automatic concept discovery from parallel text and visual corpora. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, page 2596–2604, USA, 2015. IEEE Computer Society.

[52] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[53] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[54] Y. Ushiku, M. Yamaguchi, Y. Mukuta, and T. Harada. Common subspace for model and similarity: Phrase learning for caption generation from images. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec. 2015.

[55] R. Varma, T. S. Vajaranant, B. Burkemper, S. Wu, M. Torres, C. Hsu, F. Choudhury, and R. McKean-Cowdin. Visual Impairment and Blindness in Adults in the United States: Demographic and Geographic Variations From 2015 to 2050. *JAMA Ophthalmol*, 134(7):802–809, 07 2016.

[56] R. Vedantam, C. L. Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. *CoRR*, abs/1411.5726, 2014.

[57] S. Venugopalan, L. A. Hendricks, M. Rohrbach, R. J. Mooney, T. Darrell, and K. Saenko. Captioning images with diverse objects. *CoRR*, abs/1606.07770, 2016.

[58] S. Venugopalan, L. A. Hendricks, M. Rohrbach, R. J. Mooney, T. Darrell, and K. Saenko. Captioning images with diverse objects. *CoRR*, abs/1606.07770, 2016.

[59] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator, 2014.

[60] C. Wang, H. Yang, C. Bartz, and C. Meinel. Image captioning with deep bidirectional lstms. *CoRR*, abs/1604.00790, 2016.

[61] Q. Wang and A. B. Chan. CNN+CNN: convolutional decoders for image captioning. *CoRR*, abs/1805.09019, 2018.

[62] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. *CoRR*, abs/1612.01051, 2016.

[63] R. Yamashita, M. Nishio, R. Do, and K. Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9, 06 2018.

[64] L. Yang, K. D. Tang, J. Yang, and L. Li. Dense captioning with joint inference and visual context. *CoRR*, abs/1611.06949, 2016.

[65] Y. Yang, C. Teo, H. Daumé III, and Y. Aloimonos. Corpus-guided sentence generation of natural images. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 444–454, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[66] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.