instaGAN
De Novo Food Blogging with Generative Models

Kyle Xiong and Jonathan King
Carnegie Mellon University
10-707: Topics in Deep Learning
kxiong@andrew.cmu.edu, jek1@andrew.cmu.edu

Abstract

In this paper, we aim to mimic the ever-popular Instagram food blog by using generative models to create realistic posts with images and captions. We explore methods for image generation (using Generative Adversarial Networks), text generation (using Recurrent Neural Networks), and image captioning (using Convolutional Neural Networks). We also compare the captioning effectiveness of repurposing learned features from the image generative model versus features from a convolutional network.

1 Introduction

There has been a large amount of work in the field of image captioning where, given an image, a model can reasonably describe its contents (You et al., 2016). This makes generating image/caption pairs from Instagram food blogs seem like a straightforward task. Unfortunately, (and we speak anecdotally here,) food bloggers rarely create image captions that are very descriptive of the image contents. In fact, Instagram captions often contain personal comments, mentions of other users, "hashtags", quotes, and and other ramblings. All is not hopeless, however. According to one study (Giannoulakis and Tsapatsoulis, 2016), about $2/3$ of the time, users can accurately guess the first four hashtags of an Instagram post. For our purposes, this means that while we have no guarantee that an Instagram image's caption is purely descriptive of its content, we do have proof that hashtags and content are closely related. Therefore, it is feasible that a neural network could learn some association between images from Instagram and their captions. Motivated by our own lack of culinary creativity and an obvious dearth of food blogs on the internet[1], we introduce a method called *instaGAN*, a deep neural network that utilizes a generative adversarial network, convolutional neural network, and language model to produce both realistic images and captions for an Instagram food blog. In particular, we study a novel way of integrating learned embedded features from the generative network into the image captioning model.

2 Related Work

Image generation

In the domain of image processing, generative adversarial networks (GANs) (Goodfellow et al., 2014) have been used to generate novel data points after learning from a given training set. Among the many applications of GANs are novel image generation and text to image mapping. A previous method, DCGAN (Radford et al., 2015), is a GAN based off of convolutional neural networks (Schmidhuber, 2015) which have been shown to be more successful than multi-layer perceptrons in image recognition tasks (Driss et al., 2017). DCGAN was moderately successful at generating food images and could

[1]Citation needed.

recreate the textures of food. However, it was difficult to identify the exact kind of food in the images. Furthermore, it was apparent that some modes in the GAN diverged towards outputs that the discriminator could not distinguish from real images but were obviously not images of food (see Results). Furthermore, when extended to produce outputs of higher resolution images (i.e. 256×256), DCGAN quickly becomes unstable. DCGAN's generator loss diverges to greater values, producing nonsensical images, while its discriminator becomes progressively more adept.

To our knowledge, few GAN architectures currently exist that can generate high-quality and high resolution images. One such method that does generate high-fidelity high resolution images is BigGAN (Brock et al., 2018), which the authors claim is viable for images of up to 512×512 resolution. However, the authors of BigGAN state that the model undergoes training collapse, which necessitates early stopping. Given the complexity of food images and the potential need to train a GAN for a huge number of iterations, we decided not to use the BigGAN architecture.

Instead, we opted to derive our method from another recent architecture published by NVidia, ProGAN (Karras et al., 2017), which uses progressive growing of their generator (G) and discriminator (D) networks to achieve realistic images at very high resolution. At a high level, the model starts by training the G and D networks at 4×4 resolution in generation 1. Once training is finished in generation 1, generation 2 begins by stacking an 8×8 layer on top of the 4×4 layer in both G and D. Progressive training and growing is repeated until the model reaches its target resolution.

Image Captioning

On their own, recurrent neural networks (RNNs) have been shown to be highly practical for generating text (Sutskever et al., 2011). When combined with Convolutional Neural Networks (CNNs), models can effectively learn how to embed images and subsequently generate text conditioned on these embeddings (Vinyals et al., 2016). Many of the most successful models combine these methods with attention, such as the paper *Show, Attend, and Tell* (Xu et al., 2015). These models allow the decoder to attend to different portions of the input image when generating text and lead to better caption quality.

While there has been some work on using GANs to directly produce image captions (Dai et al., 2017), to our knowledge there has been no published work on generating image/caption pairs that explicitly uses a portion of the GAN network to embed images in lieu of a traditional CNN.

3 Methods

We split up our goal of a generated food blog into several distinct tasks, described below: **3.1)** developing a training set, **3.2)** generating images *de novo*, **3.3a**) generating text *de novo*, **3.3b)** generating captions conditioned on images, and **3.4)** combining each network to generate realistic image/text pairs.

3.1 Acquisition and Pre-processing of Training Data

We used a tool called Instaloader to download the images, captions, and metadata from a wide array of food-related hashtags and user profiles that were chosen by inspection and popularity.

Image Pre-processing

In our initial experiment to train DCGAN, we scaled the resolution of all images down to 64x64 and stored approximately 100,000 training images as a PyTorch tensor. To train ProGAN to generate much higher resolution images, we resized all images down to 256x256. Because there were images that clearly were not food, we also used a CNN to filter out any non-food images. In total, 115,360 training images were stored in a separate folder on a hard drive because they wouldn't all fit on memory if loaded simultaneously. To test ProGAN with different hyperparameters, we used smaller datasets which were quicker to train on but less realistic than using a much larger dataset, we stored datasets of up to 15,000 images as PyTorch tensors.

Text Pre-processing

To train the text generation and image captioning models, image and caption pairs previously downloaded were screened to exclude non-English captions using a port of Google's language detection software for Python called `langdetect`. Furthermore, for uniformity and to decrease model complexity, many character such as brackets, quotes, and symbols were normalized to a single symbol (i.e. all types of curly quotes became "). Our model allows for 1137 distinct characters including Emojis, popular symbols such as bullet points, and all English characters. Unknown characters have been replaced by the backtick symbol. The start and end-of-caption characters are set as "\", and "\n", respectively. Newline characters within captions are substituted with tab characters.

3.2 Generating Images *de novo*

Training DCGAN

We trained our baseline method, DCGAN, on approximately 100,000 training food-related images for 550 epochs with a batch size of 32. We used the same hyperparameters and optimizer as DCGAN - Adam (Kingma and Ba, 2014) optimization with a learning rate of 0.0002 and betas of 0.5 and 0.999. However, we noticed that with the Instagram dataset, DCGAN had a strong tendency to overfit such that the discriminator learned much faster than the generator. Eventually the discriminator's loss reached zero and the generator stopped learning how to "trick" the discriminator.

To improve on DCGAN in our application, we applied a dropout (Srivastava et al., 2014) of 0.5 to the activations of each convolutional and deconvolutional layer. Discriminator loss decreased more slowly and we could train for more epochs, but discriminator loss still steadily decreased, which still limited the number of epochs we were able to train DCGAN for.

Training ProGAN

We initially trained ProGAN on 14,000 images from the hashtag #foodstagram, which included mostly food-related images upon inspection. #foodstagram alone was chosen because training ProGAN and selecting the right hyperparameters would be faster using a smaller training set loaded directly into system memory at first. We later expanded our training set and filtered out non-food images by using Inception V3's ImageNet CNN model to classify images in all hashtags and including images in the training set only if their most likely class belonged to an ImageNet food class. Images were then resized to 256×256 and stored in a separate folder. We were able to find 115,360 food images and used PyTorch's ImageFolder dataset to load images from a new root folder that contained all training images.

A ProGAN model with a latent layer of 256 was trained for 10, 20, 20, 20, 20, 20, and 20 epochs, using batch sizes of 32, 32, 32, 32, 32, 32, and 16 and fade-in percentages of 50 for image resolutions 4×4, 8×8, 16×16, 32×32, 64×64, 128×128, and 256×256 respectively. Batch size was lowered to 16 for the final resolution because the amount of memory gradient computation requires at that resolution is more than was available on our GPU (11 GB). A latent space of 256 was chosen for its relatively short training time and its lower GPU memory cost compared to larger latent spaces.

3.3 Caption Generation

With the goal of image captioning in mind, we decided on a modular model architecture for text generation and image captioning that, at an abstract level, contains an encoder and decoder. The starting code was forked in Github from a PyTorch tutorial on image captioning written by Yunjey Choi. In our project, we have kept the general framework from Choi but have added our own model definitions, as well as functionality for pre-training the ability to utilize features from a GAN model. When trained for text generation, both the encoder and decoder are RNNs, allowing the model to learn the semantics of Instagram captions, Figure 1, left. Later, the encoder RNN can be substituted with an encoder CNN, at which point the model may learn the mapping between the image and its caption, Figure 1, right in orange. Alternatively, the ProGAN's discriminator can be used as an encoder in lieu of a CNN by extracting an embedding from its final layer, Figure 1, right in blue.

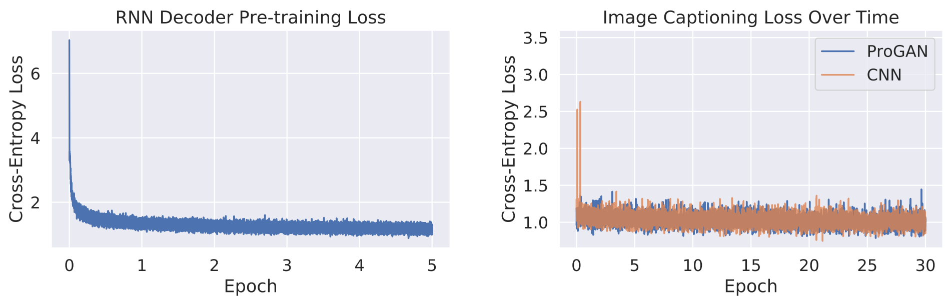


Figure 1: The image captioning model's training loss when pre-training on text with a RNN $\rightarrow$ RNN architecture, **left**, and while training on images and text with the $\{$ProGAN, CNN$\}$ $\rightarrow$ RNN architecture, **right**.

Generating text *de novo* (RNN $\rightarrow$ RNN)

In order to teach the Decoder RNN how to effectively generate text, we train the encoder-decoder model on text alone following common practices for language models. In this instance, both the encoder and decoder are 4 layer Gated Recurrent Units (GRUs) trained on size-16 batches of padded, variable-length sequences. Each GRU has 1024 hidden units, accepts 512-dimensional character embeddings as input, and produces softmax-normalized 1137-dimensional output vectors, representing character probabilities. The encoder's final output is fed into the decoder model, followed by a special start-of-caption character. Then, for each character in the training sequence, the decoder is trained to predict the next character using cross-entropy loss. Because empirical evidence of RNN models' difficulty to generalize to long sequences (Bai et al., 2018), the captions are split up into 75-character long segments when training. The text generation model was trained for 5 epochs on a corpus of more than 695,000 pre-processed English Instagram food blog captions which, when split into 75-character chunks, yielded several million training examples. The model was optimized using the Adam optimizer (Kingma and Ba, 2014).

Generating text conditioned on images (Image $\rightarrow$CNN $\rightarrow_{embed}$ RNN)

In our first attempt to caption images, we utilize the same general encoder-decoder architecture as in the text generation model. However, instead of the input to the decoder being an embedding from an RNN, we utilize a pre-trained CNN (Resnet-152 (He et al., 2015)) to generate image embeddings. The RNN Decoder is initialized using the pre-trained weights learned from the text-only model. During training, the CNN's weights are fixed and the rest of the model is trained similarly to the text-only model using cross-entropy loss. The learning rate for this model is initialized to 10^-6 to account for the fact that the RNN Decoder has already been pre-trained.

Generating text conditioned on images (Image $\rightarrow$ ProGAN $\rightarrow_{embed}$ RNN)

In our second attempt to caption images, we devised a way to incorporate learned features from the trained ProGAN model. We hypothesize that one of the layers of the ProGAN's discriminator may be used effectively in lieu of a separate CNN model to create an image embedding. Intuitively, the ProGAN should have learned to recognize certain features of each image which could be used to inform the RNN Decoder for caption generation. To output embeddings of images input to the GAN's discriminator, we copied the output from the discriminator's final convolution layer and output it in addition to the scalar discrimination result. In order to not influence the training process, we modified the discriminator's forward function to output the embedding only when a boolean `output_embeddings` flag is set to `True`. During the ProGAN model's training, the flag is set to `False` by default so that the discriminator's forward function only outputs the scalar discrimination result.

The ProGAN $\rightarrow$ RNN model is trained in the same way as the CNN $\rightarrow$ RNN model, with the only exception being that, due to the fact that the ProGAN's embedding is 4096-dimensional, an additional linear layer is trained to transform the 4096-embedding into the 512-dimensional embedding expected

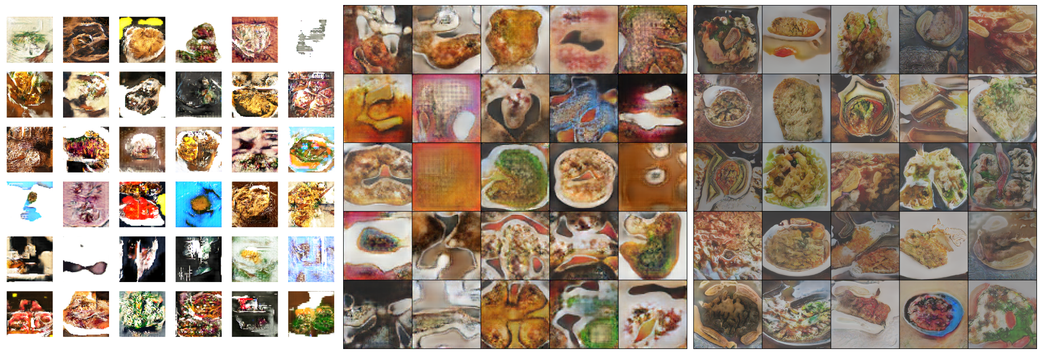


Figure 2: **Left**: A random sample of 36 64x64 images from DCGAN when trained on 100,000 images across multiple hashtags. **Middle**: 25 samples of 64x64 images from ProGAN when trained on #foodstagram without a CNN filter. **Right**: 256x256 images from ProGAN when trained on 15,000 food images from multiple hashtags.

by the RNN Decoder. The learning rate for the linear layer is set at 10^-4 compared to 10^-6 for the rest of the model to account for the fact that it has not been pre-trained.

4 Results

DCGAN Generates Moderately Successful Images of Food

Shown in Figure 2 (left), DCGAN generates images of what we can perceive as food. In some images, DCGAN was able to reproduce textures of real food. However, upon closer inspection, it's difficult to ascertain exactly what food items were being generated. While DCGAN was able to capture the texture of food, it seems that its discriminator doesn't give much weight to the actual shape of the food. In addition, some of the generated images were largely white blotches with small patches of indistinguishable textures. This suggests that DCGAN still overfits after a large number of epochs.

While DCGAN is mildly successful at 64×64 resolution, we were unsuccessful at training a modified DCGAN to output 256×256 images. We can improve by training ProGAN on larger images.

ProGAN Generates More Successful Higher Resolution Images of Food

Using a smaller training set of approximately 14,000 images from the #foodstagram hashtag, we found that our images were unintelligible (Figure 2, left). We only trained up to 64x64 resolution because the generator's loss was already far greater than the discriminator's loss, with no trend of decreasing further. While we achieved more success when increasing the size of the latent space to increase the number of trainable parameters, this tweak ended up consuming too much VRAM. We attributed the initial unintelligible results to two possibilities - our training was too small and/or the food images themselves had too many non-food objects.

After we applied Inception V3 to filter out non-food images (see Methods), we got somewhat better results and we could see that some of the generated images were reflective of real food (Figure 2, right). However, the contrast appeared to be very poor, as if a gray filter was overlayed on top of the generated images.

We then trained a ProGAN model on a much larger dataset consisting of 115,360 filtered food images (see Methods). After training, we saw that there was a noticeable increase in image quality compared to DCGAN and ProGAN trained on a smaller dataset (Figure 4).To quantify the quality of images generated by our trained ProGAN model, we computed the Inception Score (IS) and Frechet Inception Distance (FID) of images generated by ProGAN and DCGAN (Table 1). FID was computed by inputting 5,120 GAN-generated images and comparing the distribution of these images to the distribution in the training set. Both IS and FID were noticeably higher for ProGAN at 256x256 resolution compared to DCGAN at 64x64, suggesting significantly more realistic generating images even when generating high resolution images where the margin of error is far higher for generative networks.

Metric	IS	FID
Real Images	11.37	4.49
ProGAN	5.87	57.53
DCGAN	4.01	93.18

Table 1: Metrics to quantify and assess the realism of ProGAN and DCGAN's generated images - inception score (IS) and Frechet Inception Distance (FID).

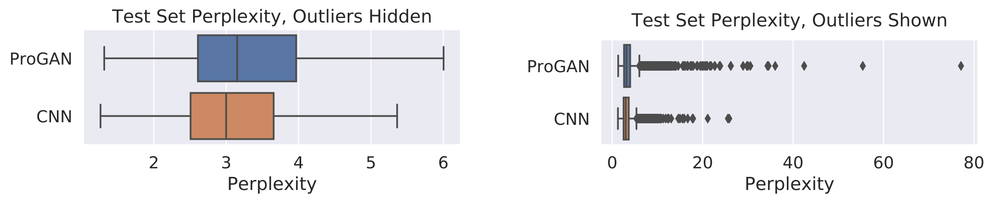


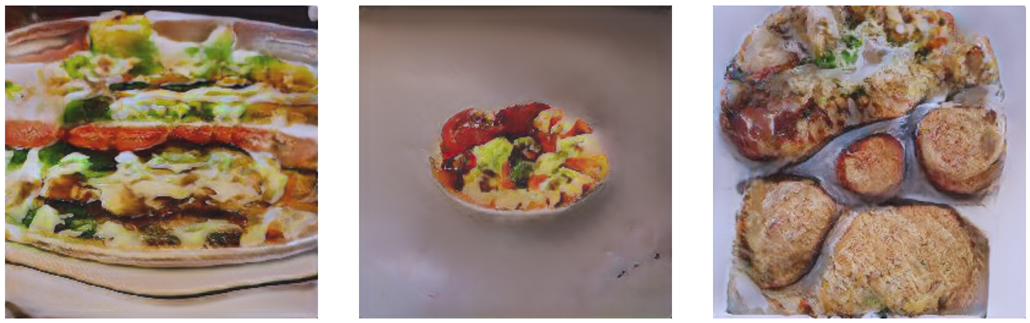
Figure 3: Comparison of image captioning model performance (perplexity per caption) on a test set when images are embedded with ProGAN vs. CNN.

Image Captioning: Image $\rightarrow$ {ProGAN, CNN} $\rightarrow_{embed}$ RNN

Figure 1 demonstrates that both the ProGAN and CNN models achieved similar values for Cross Entropy loss during training. In order to better compare the performance between using the ProGAN and CNN image embeddings in the image captioning model, we constructed an independent test set of around 7,000 images collected and pre-processed in exactly the same way as the training set. We embedded each image with either the ProGAN or CNN model, and then captioned it using the same decoder, measuring the model's perplexity in each instance. Figure 3 reports the average Perplexity per caption for both the ProGAN and CNN embedding image captioning models. The ProGAN and CNN embedding methods work very similarly, while the ProGAN has many more outlier captions with very high perplexity. On average, the CNN embedding method is marginally better.

Final model: ProGAN $\rightarrow_{generate}$ Image $\rightarrow$ CNN $\rightarrow_{embed}$ RNN

We observe that our final trained image captioning model is able to make quasi-realistic Instagram-style captions for each image depicted. Every caption ends with several newline characters, symbols, and food-related hashtags. Emjoi characters are gently sprinkled at appropriate locations in the



Homemade Protein on Saturday lunch? .
.
.
.
.
.
.
.
.
#pizzaporn #photo #pic#sushi #restaurant #bread

Can't get more time to start the weekend!! Super creamy low-carb purée... chocolatey Chocolate and Grape ice cream biscuits available for lunch today.🍉😍•v
•
#foodbeast #eeeeeats #eatfamous #fwx #buzzfeast #feedfeed #eater #yougottaeatthis #bloglovinfood #beautifulcuisines #seafoodporn #foodforthought #funinthephickendinner `#eatstagram #eattheworld #foodstatic #huffposttatas

For those who love these protein waffles....
Credit @preview.app
.
Follow @thegumansingharbor to!
📸: @dark_aiors
#highprotein #latergram #seafood #spantain #food #ourplateblogger #101eats #brunch #aanderson #avocadotoast #seagoldens #London #thisiswhyimfat #nycfoods #yyclovesfood #yorkfood #rrstrael #nycfood #caloriecountyupgun #hungrywoman #handsinframe #foodpornography

Figure 4: Sample images generated by ProGAN with captions generated by our CNN captioning model after training on a large dataset, filtered to only include images of food with English captions.

post. Sometimes, posts include links to non-existent users, URLs, or poorly formatted recipes. Unfortunately, despite these favorable qualities, captions are rarely very coherent. The authors argue, however, that the coherency of these generated posts is not much less than that of actual posts. One of the author's advisor (who holds a Ph.D. in Computer Science from Carnegie Mellon University) was even fooled by one such generated caption.

Figure 4 contains several examples from our complete model that were hand selected to demonstrate different behaviors. In the leftmost example, both the image and caption are relatively generic looking, which increases the plausibility of the post. However, in the middle example, we notice that the image is more clearly a bowl of brightly colored food, such as fruit or salad. However, in this case, the caption claims that it contains chocolate, and "grape ice cream biscuits", which is neither correct nor plausible. In the third case, we see some agreement between the caption "protein waffles" and the image, which possibly includes something doused in syrup. In all, given the lack of clearly differentiated food types, the model's similarly vague and incoherent captions seem to plausibly describe the generated images.

5 Future Work

We believe that the ProGAN model could produce better results if we further partition the food dataset into their respective classes and use a conditional ProGAN. However, given that there are about 40 different food classes, we would ideally need an even more massive dataset so that each food class is properly represented. Such a task is not impossible, but if we assume that each class needs at least 50,000 images for conditional ProGAN to produce realistic images, we would need to expand our dataset by about 20 times. Furthermore, a conditional ProGAN trained on such a dataset would take about 20 times longer to train than the current model, which already takes around 4 days to train on a GTX 1080 Ti.

Without the time limit imposed by 10-707, we can further improve instaGAN by collecting even more Instagram photos, filtering them with Inception V3, and partitioning them by food class. We can then train a conditional ProGAN on the much larger dataset using the same hyperparameters. We believe that such a model will produce much better class-specific images as each class's training images should be far more homogeneous.

Furthermore, we can better utilize post metadata to select for the most popular and potentially highest quality content (i.e. professional photos, captions with meaningful content, etc.).

Currently, the ProGAN model and RNN are trained separately and information from ProGAN is fed forward into the RNN decoder, but not the other way around. In the future, we could incorporate a strategy to train models simultaneously and allow a gradient from the RNN to be passed back into the GAN to assist image generation.

References

S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018. URL `http://arxiv.org/abs/1803.01271`.

A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018.

B. Dai, S. Fidler, R. Urtasun, and D. Lin. Towards diverse and natural image descriptions via a conditional gan. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2970–2979, 2017.

S. B. Driss, M. Soua, R. Kachouri, and M. Akil. A comparison study between mlp and convolutional neural network models for character recognition. In *Real-Time Image and Video Processing 2017*, volume 10223, page 1022306. International Society for Optics and Photonics, 2017.

S. Giannoulakis and N. Tsapatsoulis. Evaluating the descriptive power of instagram hashtags. *Journal of Innovation in Digital Ecosystems*, 3(2):114 – 129, 2016. ISSN 2352-6645. doi: https://doi.org/10.1016/j.jides.2016.10.001. URL `http://www.sciencedirect.com/science/article/pii/S2352664516300141`.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

I. Sutskever, J. Martens, and G. E. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge. *CoRR*, abs/1609.06647, 2016. URL `http://arxiv.org/abs/1609.06647`.

K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015. URL `http://arxiv.org/abs/1502.03044`.

Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image captioning with semantic attention. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.