<u>VISUAL RECOGNITION</u>

MINI PROJECT REPORT

Unique Group Code: AGMK

Team member details:

Name: Anish Rajan Roll No.: IMT2018009
Name: Gaurav Goel Roll No.: IMT2018025
Name: Krishna Phalgun Roll No.: IMT2018038
Name: Manasa Kashyap Roll No.: IMT2018040

Problem:

Design a CNN-LSTM system (preferably in pytroch) that can perform image captioning.

Constrain: The total model size should not exceed 100 MB. Considering that each parameter weight is a float value (4 bytes), your CNN-LSTM model should have approximately 25 Million parameters or less.

Introduction:

Image Captioning refers to the process of generating captions of the image based on the objects and actions in the image. It uses Computer Vision and Natural Language Processing concepts to recognize the context of an image and describe them in a natural language like English. The dataset will be in the form (image, captions). The model to generate these captions follow an encoder-decoder architecture which uses an abstract image feature vector as an input to encoders. Object detection models, Convolutional Neural Network (CNN) and attention-based Recurrent Neural Network have helped a lot to improve the image caption results.

Model Architecture:

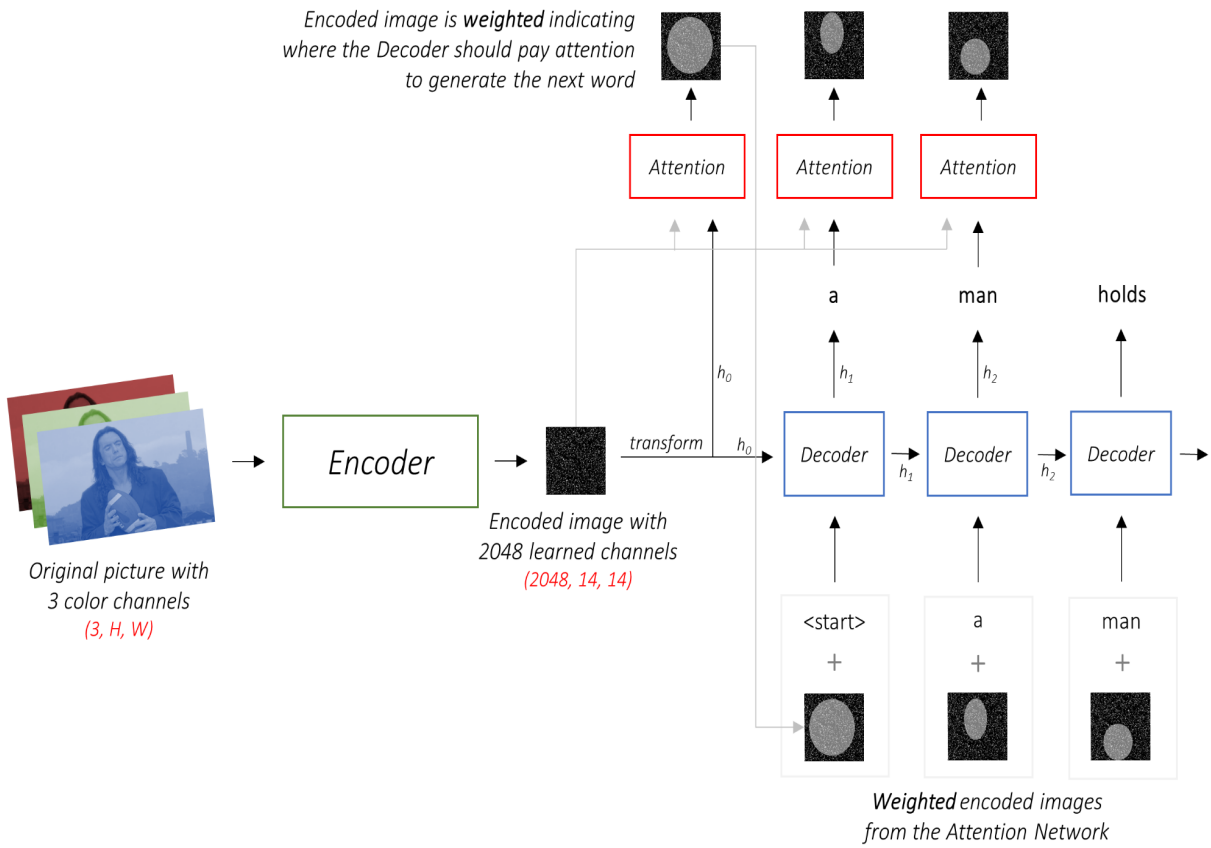


Figure1 - Model Architecture

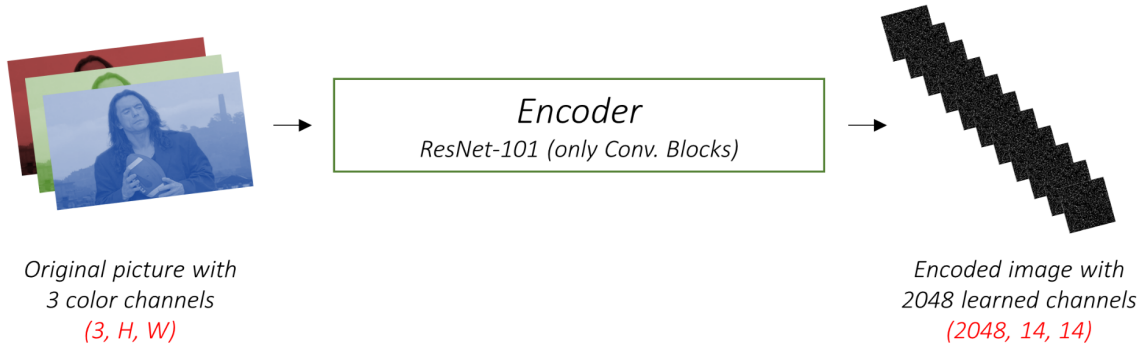


Figure2 - Encoder

As an Encoder, a pre-trained Convolutional Neural Network model - MobileNet v2 with **2223872 parameters** is used. In contrast to traditional residual models, which use expanded representations in the input, it has an inverted residual structure, with the input and output of the residual block being thin bottleneck layers. It filters features in the intermediate expansion

layer with lightweight depth-wise convolutions. In order to maintain the representational power, non-linearities in the narrow layers were also removed. The smaller encoded output represents a summary of the usefulness of the image. Each subsequent representation is more "learned," with a greater number of channels, as the model creates smaller and smaller representations of the original image. The final encoding has a 14x14 resolution and 1280 channels. To resize the encoding to a fixed size, the AdaptiveAvgPool2d layer is used.

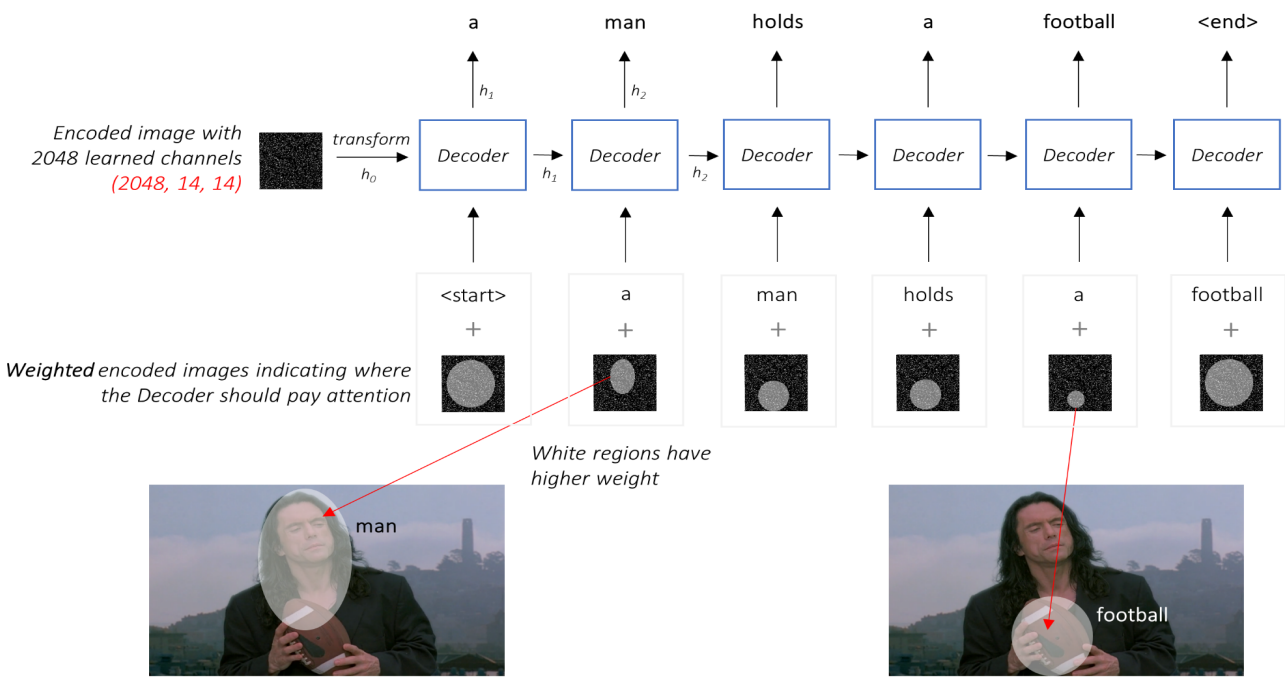


Figure3 - Decoder with Attention

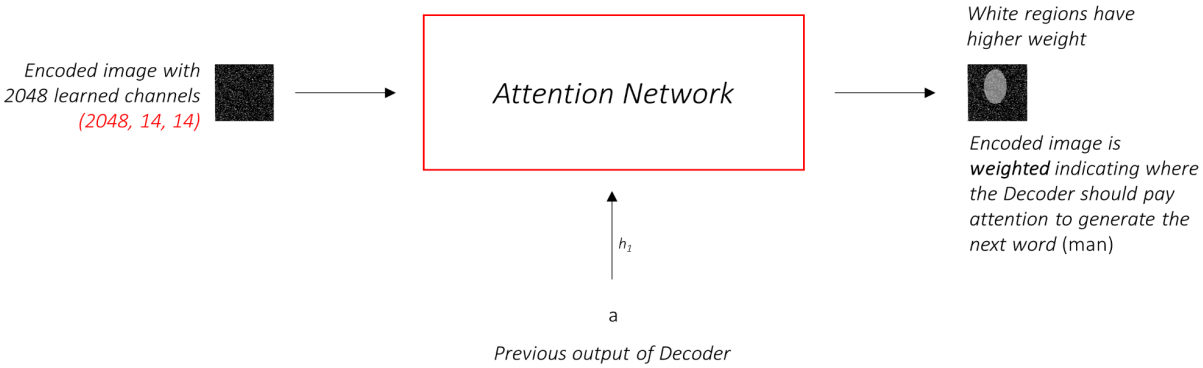


Figure4 - Attention

As a Decoder, a Recurrent Neural Network with a stack of LSTM layers and an Attention Mechanism with **12250288 parameters** is used to generate a caption word by word. Only linear layers and a couple of activations make up the Attention network. Both the encoded image (flattened to N, 14 * 14, 1280) and the hidden state (output) from the Decoder are transformed to the same dimension, namely the Attention size, by separate linear layers. The hidden and cell state of the LSTM is initialized using the encoded image using two separate linear layers. They are then added and ReLU activated. The result is transformed to a dimension of 1 by a third linear layer, after which the softmax is used to generate the weights alpha. A sigmoid activated linear transform of the previous hidden state Decoder filter or gate is used to process the attention-weighted encoding to put more emphasis on the objects in the image. To generate the new hidden state, the LSTMCell is concatenated with this filtered attention-weighted encoding and the embedding of the previous word. This new hidden state is transformed into scores for each word in the vocabulary by a linear layer, which is then saved. The model looks at various parts of the encoded image at different points in the sequence. The weighted average across all pixels is used instead of the simple average, with the weights of the important pixels being higher. To generate the next word, this weighted representation of the image can be concatenated with the previously generated word at each step.

A linear layer is used to transform the output of the Decoder into a score for each word in the vocabulary. Beam search is used to find the best caption. It expands all possible next steps and keeps the k most likely, where k is a parameter that controls the number of beams or parallel searches through the sequence of probabilities, rather than greedily choosing the most likely next step as the sequence is constructed.

Maximum Likelihood Estimation was used to train the model using cross-entropy loss. At each time point during training, words of the given sentence are fed, and the expected output is the next word in the sentence. As a result, the training protocol seeks out the best parameters for explaining text data. Instead of feeding in the true words, the output of the current time step is used as the input for the next one during inference. Over the padded regions, the losses are not computed. The Adam

optimizer is used to train the model for 20 epochs with a learning rate of 1e-4.

Automated Bilingual Evaluation Understudy (BLEU) evaluation metric is used to assess the model's performance on the validation set. This function compares a generated caption to a reference caption(s). All captions available for an image are used as reference captions for each generated caption.

Training Methodology:

The data set consists of 8,000 images that are each paired with five different captions which provide clear descriptions of the salient entities and events. Since a pre-trained Encoder was used, the images were pre-processed into the format this pre-trained Encoder is accustomed to. The Encoder requires three inputs.

1. Images:

   The images were split into mini-batches, each of which contained three-channel (RGB) images of the shape (3 x H x W), with H and W, set to 256. The images were scaled to a range of [0, 1] before being normalized with mean = [0.485, 0.456, 0.406] and standard deviation = [0.229, 0.224, 0.225].

2. Captions:

   Since each word is used to produce the next word, captions are both the target and the inputs of the Decoder. The zeroth word, <start> is used to generate the first word. The last word <end> is used to denote the end of a caption. The Decoder is trained to predict it to know when to stop decoding during inference.

3. Caption Lengths:

   The lengths of each caption were tracked because the captions are padded. This is equal to the length of the actual caption plus two

(for the start> and end> tokens). To avoid wasting compute on the pad tokens, each sequence is only processed up to its length.

The images were fed to the model as Float tensors of dimension (N, 3, 256, 256), the captions as Int tensors of dimension (N, L) where L is the padded length, and the caption lengths as Int tensors of dimension (N,) where N is the batch size.

The N images and captions are sorted in order of decreasing caption lengths at the beginning. This is done to ensure that only true time steps are processed, rather than the pad tokens.

Training Level Experiment Details:

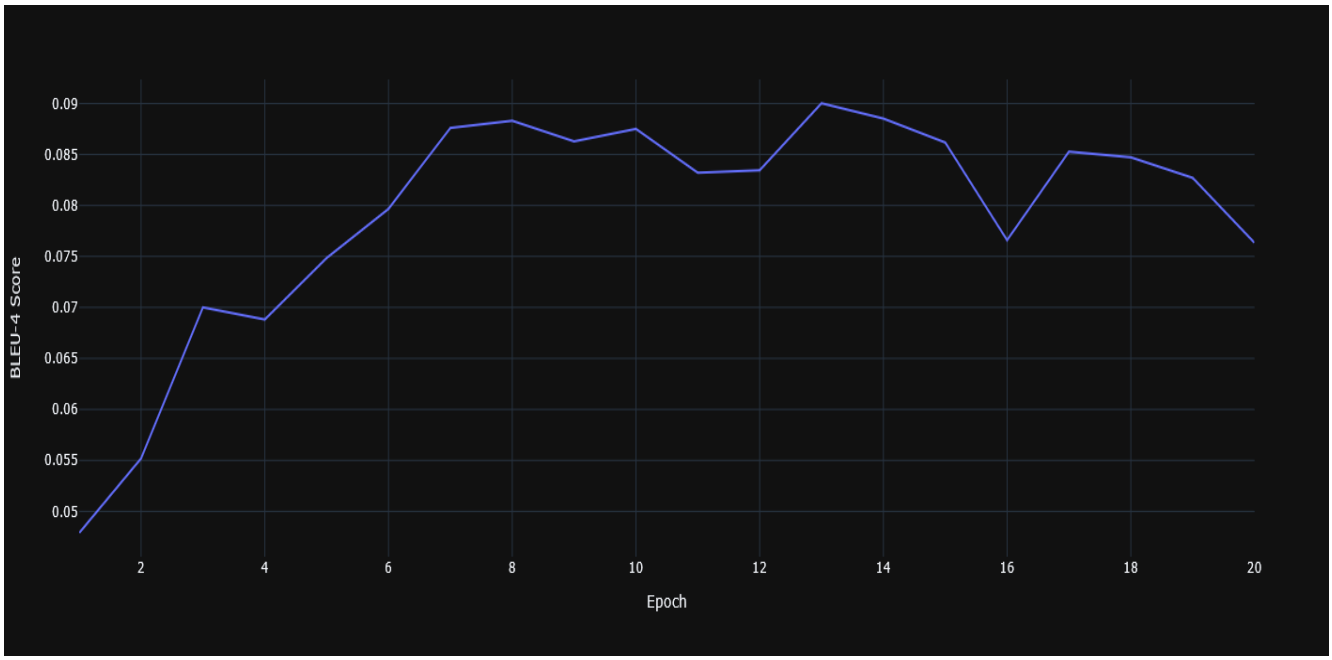
In order to improve performance during model training, the following experiments were carried out.

1. Fine Tuning the Encoder:

   The MobileNet-v2 model tends to require more computational resources to update the weights of the two convolutional blocks, so it is not fine-tuned since the convolutional blocks must have learned something very basic to image processing, such as detecting lines, edges, curves, and so on.

2. More Epochs:

   The model was trained for 5 epochs at first, and then the number of epochs was increased by 5 until it reached 20. A plot of validation BLEU-4 scores over the training epochs is shown below. At the 13th epoch, the validation BLEU-4 score peaked at about 0.0900183851042128. This is the corpus_bleu score of the whole validation set from nltk which is different from the sentence_bleu score. The sentence_bleu score at the peak was 0.449827474626.



3. Increasing LSTM Layers:

   More LSTM layers were introduced to allow for greater model complexity and access to more levels of abstraction of input observations over time, but this posed a challenge to model specifications and significantly increased computational complexity. As a result, the model only has one LSTM layer.

4. Teacher Forcing based Training:

   This is a training methodology used for faster convergence. When the dataset is small, it is hard to train a model from scratch and it may take a lot of time to converge. In this kind of training, at each step of the decoder, ground truth is fed as the next input instead of the previous output. In this way, the model will be able to converge faster. We have experimented with different kinds of teacher forcing ratios. The best teacher forcing ratio was 0.5 in which the model was able to learn properly and converge faster. But, the drawback is that the inference has to be written separately which we have done.

Inference:

For inference from the model, we have implemented a beam decoder. This is a decoding method in which at each time step of the decoder, top-k scores that can be output are maintained. The next time step would have k*k outputs and only top-k is taken from them. This is a very useful technique because the model may make a mistake at some point in time and the resulting NLL loss at the end would be more as compared to if it would have taken a different output at some previous time step. The beam size set by us was 10.

Results:

The mean sentence_bleu score on the 1000 images of the test dataset was 0.40238592. The final number of parameters used inside the model was approx 14 million. The total memory size of the encoder, the decoder was 72 MB.

Subjective Results and Predictions on Test Set:

Image 1

Prediction - a boy in a red shirt and white dog is running on a beach

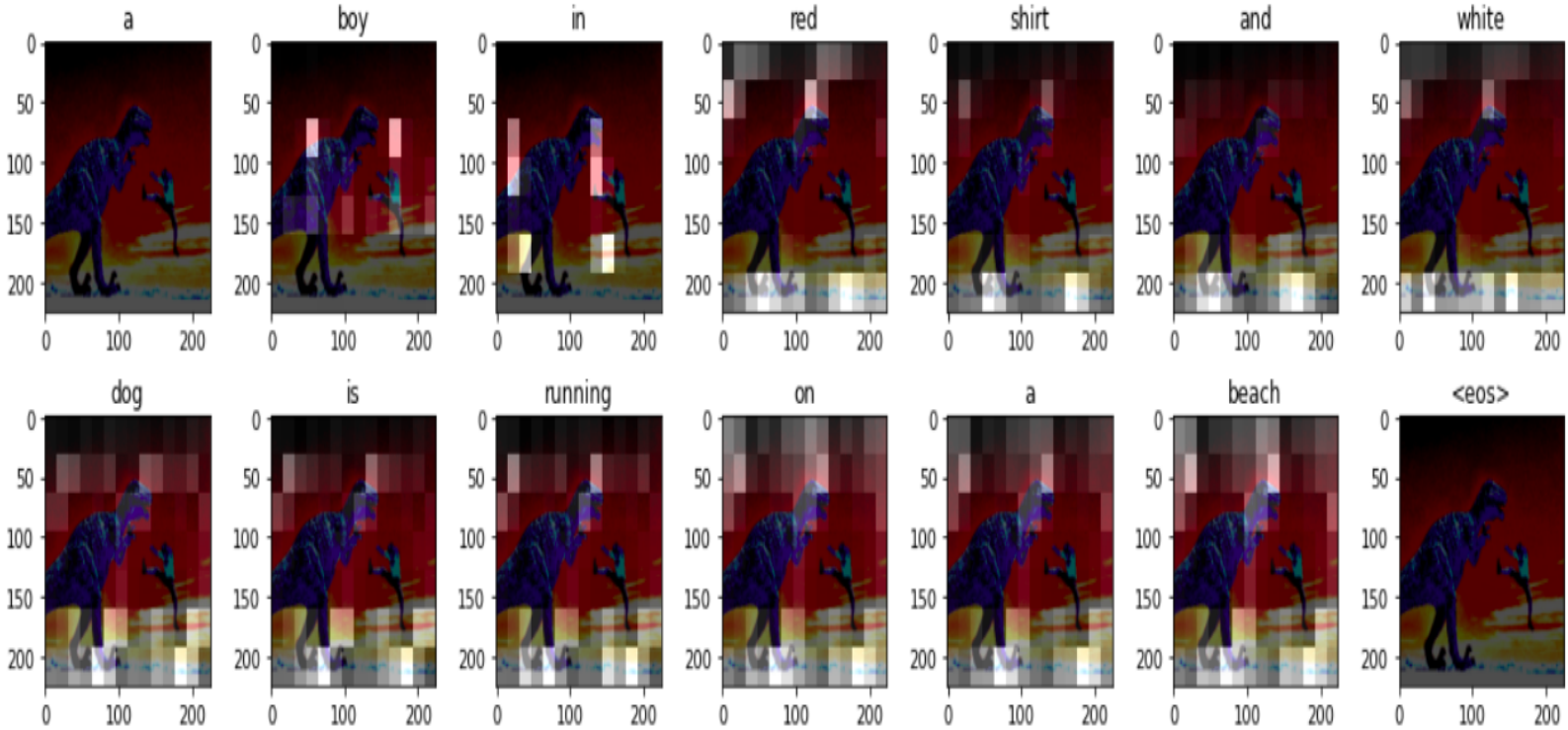


Image 2

Prediction - a man in a blue shirt and jeans sits on a bench.

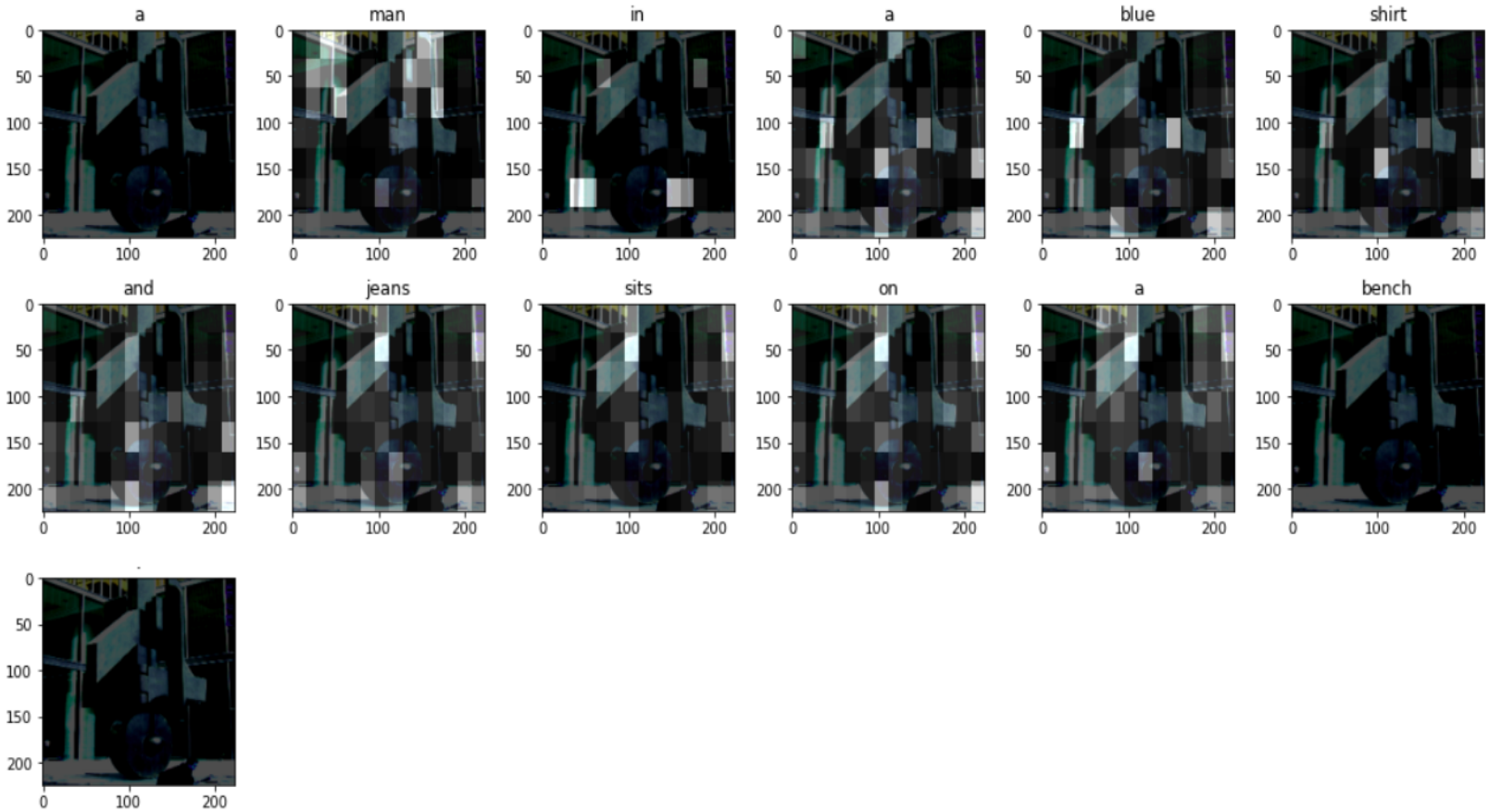


Image 3

Prediction - a group of people stand in the street.

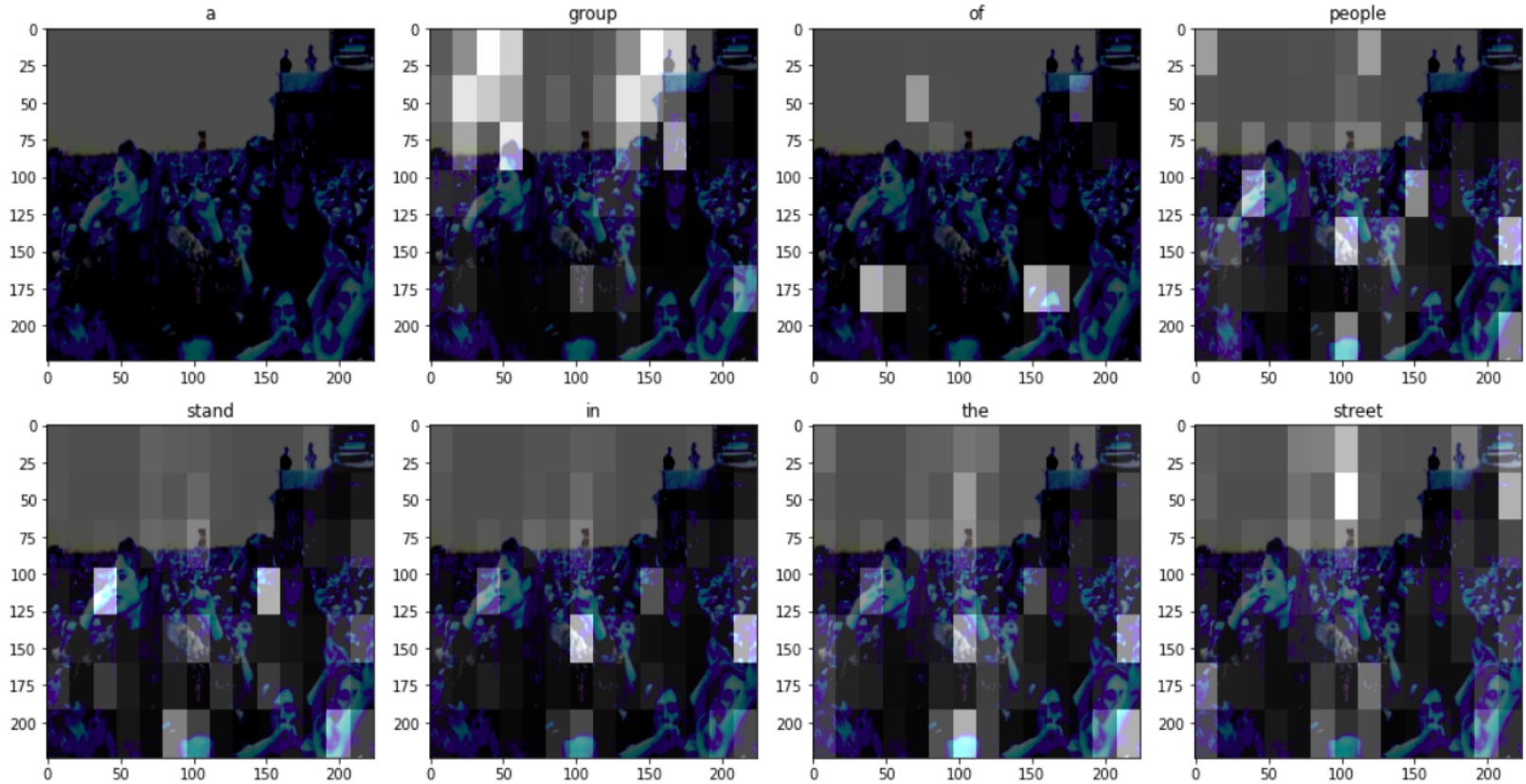


Image 4

Prediction - a black and white dog jumping over a hurdle.

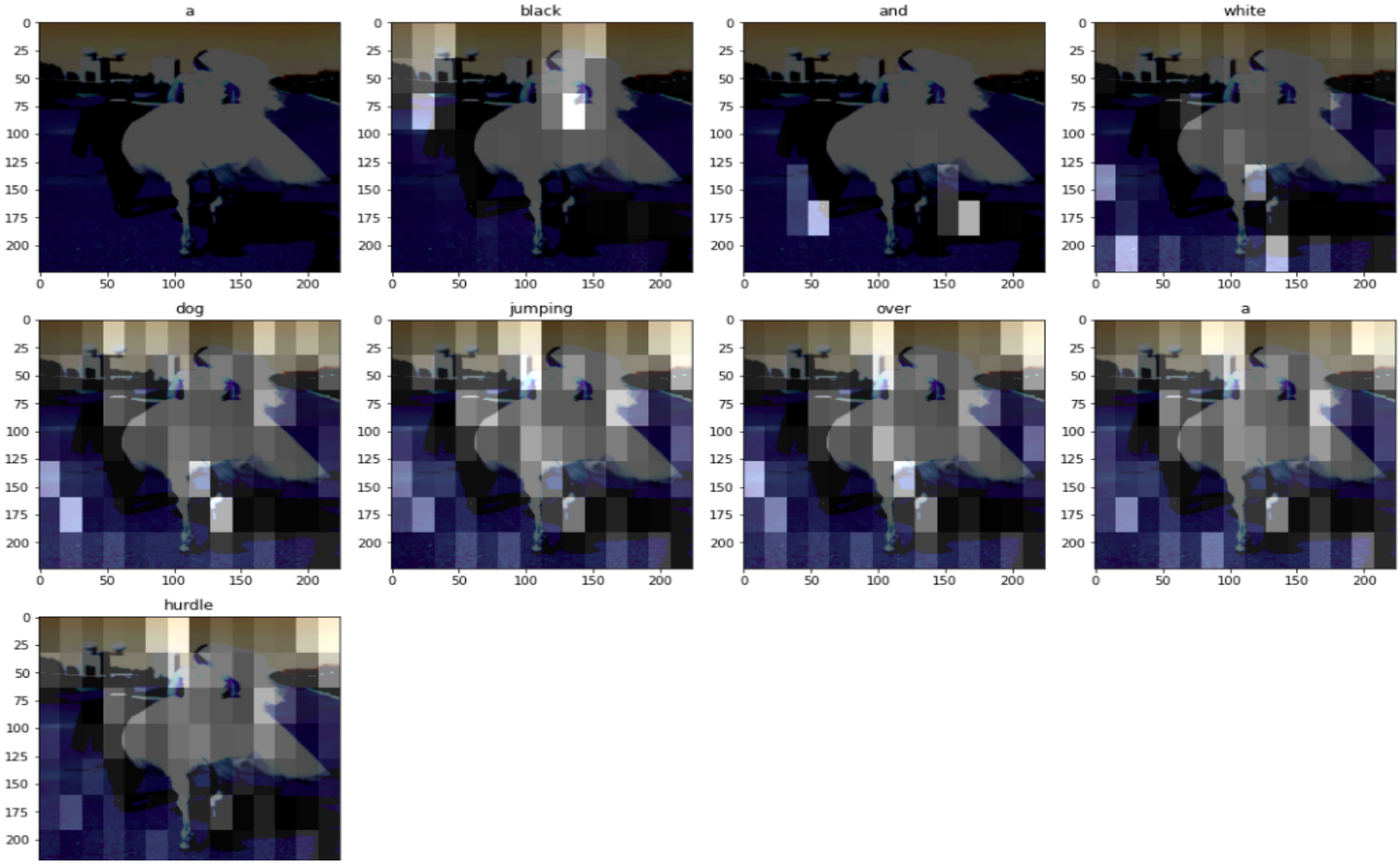
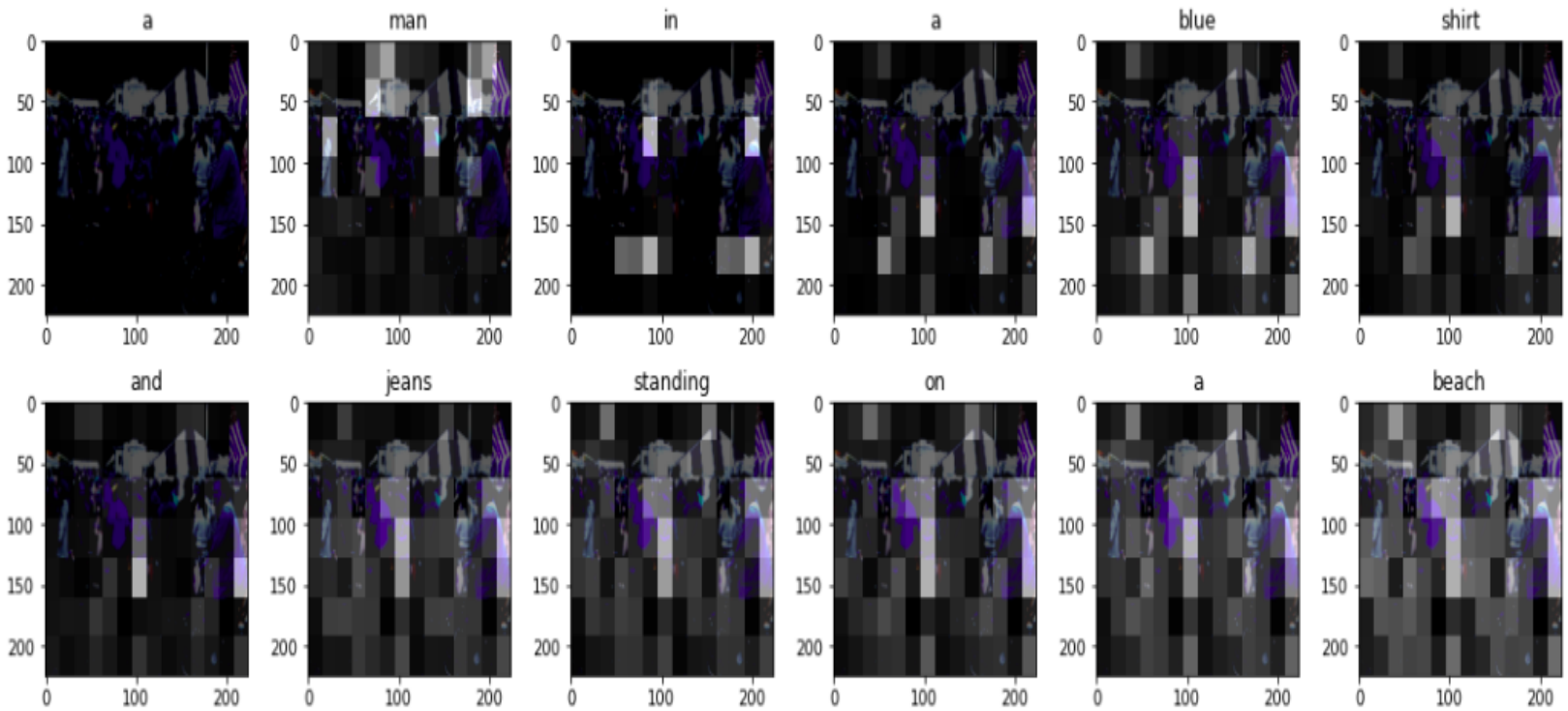


Image 5

Prediction - a man in a blue shirt and jeans standing on a beach.



References:

- https://medium.com/@raman.shinde15/image-captioning-with-flickr8k-dataset-bleu-4bcba0b52926
- https://www.nltk.org/_modules/nltk/translate/bleu_score.html
- https://machinelearningmastery.com/calculate-bleu-score-for-text-python/
- https://medium.com/@heron20012003/image-caption-by-transfer-learning-flickr8k-data-51b54d3e6e02
- https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c
- https://arxiv.org/pdf/1502.03044
- https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2
- https://data-flair.training/blogs/python-based-project-image-caption-generator-cnn/