TDDE19 Project | Group 6 | Image diversity

Anton Hansson
IDA Linköping University
antha652@student.liu.se

Dylan Mäenpää
IDA Linköping University
dylma900@student.liu.se

Lawrence Thanakumar Rajappa
IDA Linköping University
lawra776@student.liu.se

ABSTRACT

Many of today's modern applications such as social media recommend similar content to users based on what the user has previously consumed. This leads to users missing out on original content. One way of solving this is to instead recommend divergent content. In this project two approaches for finding divergent images are studied: 1) using a deep ranking model; 2) using a Xception-CNN model. The deep ranking model builds on a pre-trained convolutional neural network that is fine-tuned for the task, using a triplet of images as input. The Xception-CNN model takes a different approach, where a pre-trained convolutional Neural Network is used for generating captions for a given image. The text-similarity between generated captions is then used to find divergent images. Both models are concluded to be viable choices for learning fine-grained image similarity and diversity, that can capture both between-class and within-class image differences.

1 INTRODUCTION

Many of today's popular applications have functionality for suggesting content that is similar to what the user has already seen or liked. This tends to corner the user in a bubble and limit the content that the user is exposed to. This could be negative for the user's creativity and therefore approaches that suggest more original content for the user is of interest.

The aim of this project is to get experience regarding the task of image diversity. In this report, two different approaches that try to find divergent images have been implemented and evaluated. Like most tasks in the field of image similarity, these approaches utilize the ability of deep neural networks to extract image features. Given a query image, these models try to find the N most divergent images among a set of images. Two approaches are suggested: 1) a Deep Ranking Model (DRM) creating image embeddings; and 2) an Xception-CNN Model for generating captions of images and using text-similarity.

1.1 Deep Ranking Model

One approach to achieve a fine-grained image similarity and divergence is to employ deep ranking architecture. Deep ranking uses deep convolutional neural networks (CNNs) that interpret the relation between input image pairs, and have shown promising performance in image similarity tasks [15]. The goal is to create an image classifier that does not only consider two images to be similar if they belong to the same category, but also can recognize more fine-grained differences between images within the same class. A hypothesis is that a simplified implementation of the deep ranking image similarity model proposed by Wang et al. [15] also can find image similarities. Subsequently, the model can be used to find divergent images.

2 METHOD

2.1 Deep Ranking Model

Recently, Convolutional neural networks pre-trained on natural images have been used in many different tasks [12]. The lower layers of a CNN, tend to learn to find local features of a picture, such as edges. The higher layers find features that give more semantic meaning, such as shapes and objects [10]. These learned models can then be utilized for many different tasks through fine-tuning, e.g. classification.

In order to find image diversity, a model which have been learned to find image similarity can be employed. The idea is that the metric used to find similar images, also can be used to find dissimilar images. Wang et al. [15] propose a fine-grained image similarity model, that employs deep learning techniques called deep ranking to learn similarity metrics. The implementation improves upon a pre-trained CNN, and is proven to outperform models based on hand-crafted visual features and deep classification models [15]. In this paper, a simpler implementation of the deep ranking architecture is explored. Training CNNs from scratch requires large amounts of labeled data, but also excessive time and resources. Therefore, a transfer learning approach has been implemented, where a pre-trained network is utilized and then fine-tuned.

2.1.1 Model architecture. The architecture, illustrated in Figure 1, is a simplified version of the DRM proposed by Wang et al. [15]. The simpler implemented architecture is referred to as Deep Ranking Model Light (DRML). Inspired by DRM, DRML takes a triplet as input. A triplets consists of a query image p_i, positive image p_i^+, and a negative image p_i^-. This approach builds on the fact that the positive image is more similar to the query image than the negative image. Training three parallel multi-scale networks, as suggested by Wang et al. [15] is too time-consuming for the scope of this project, therefore only one single-scale network has been implemented. The triplets are passed sequentially to this network. The model consists of modified version of a pre-trained CNN, $f(.)$, called ResNet-18 [3]. The output of the ResNet-18 is forwarded to a linear embedding layer, $e(.)$, with a dimension of 4096. Finally, the outputs of the embedding layers are forwarded to the ranking layer.

2.1.2 Ranking layer. The ranking layer computes the model error using the triplet loss function l which is illustrated in eq. 1. Triplet loss is a ranking-based loss function that is often used to compare the similarity of images [15]. It teaches the network to produce similar feature embeddings of similar images, and different embeddings for images that differ from each other. To minimize the triplet loss during learning, the model's obedience of the ranking system is evaluated, and the model's gradients are back-propagated to the lower layers so that their weights can be adjusted.

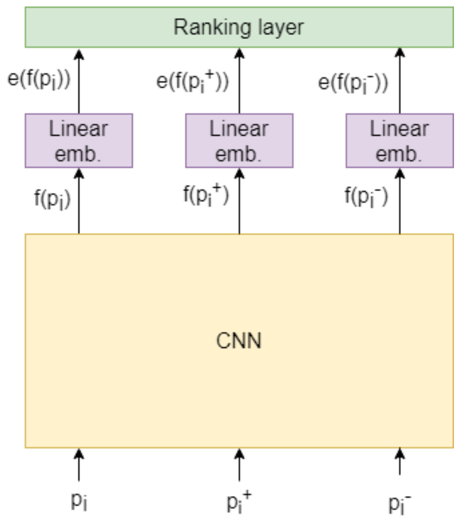


Figure 1: An overview of the triplet-based architecture

$$l(p_i, p_i^+, p_i^-) = \max\{0, g + D\big(f(p_i), f(p_i^+)\big) - D\big(f(p_i), f(p_i^-)\big)\} \tag{1}$$

D notes the Euclidean distance and g regularizes the gap between two image pairs (p_i, p_i^+) and (p_i, p_i^-). Consequently, the model is optimized so that the distance between the query image and the positive image is g lesser than the distance between the query image and the negative image. Based on results by Wang et al.[15], the parameter g is set to 1.0.

2.1.3 Tiny ImageNet 200. The organization ImageNet host an annual computer vision competition, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [1]. The competition has been developed upon a subset of ImageNet's large visual database containing more than 14 million hand-annotated images [4]. Some of today's most important deep learning neural network techniques, especially in the field of CNN, stem from this competition. For this project, the Tiny ImageNet dataset [6] was used, which is a similar challenge based on a smaller dataset of fewer image classes. Tiny Imagenet contains 200 classes. Each class has 500 training images, 50 validation images, and 50 test images. All images are of size 64×64.

2.1.4 Generating triplets. A simpler version of triplet sampling than of the original deep ranking paper was implemented. Triplets are pre-computed where paths to images are stored in a text file. Each line in the text file contains paths to the images in the order of query image, positive image, and the negative image, denoted p_i, p_i^+, p_i^- respectively.

A related triplet sampling implementation by Wang et al. [15] have negative samples that are composed of two different types of samples: in-class and out-of-class. For this project, only out-of-class samples will be implemented. Again, out-of-class samples are images sampled randomly from any class except the class of the query image. This way, many triplets can be created by using different combinations of query, positive, and negative images. Ergo, the model is trained on data where positive images are sampled from the same class as the query image; and negative images from any other class.

p_i: Input to the Q (Query) network. This image is randomly sampled across any class.

p_i^+: Input to the P (Positive) network. This image is randomly sampled from the SAME class as the query image.

p_i^-: Input to the N (Negative) network. This image is randomly sampled from any class EXCEPT the class of 'p_i'.

2.1.5 Training. The network was trained using Pytorch [11], and the implementation was designed to run on both CPU and GPU. However, memory problems may occur while handling image loading on computers without a dedicated GPU. Therefore, all training was done on a NVIDIA GeForce GTX 1080 GPU with 8GB memory utilizing the CUDA platform available in Pytorch [11].

When applying deep learning models, the dataset used for training has a great impact on the result. For example, a variation in the number of classes, class imbalance, and the total number of images used will cause different outcomes. The most common obstacle when working with CNNs is the limited amount of training images. To address this problem and reduce the risk of overfitting, image augmentation was applied to all training images. This was done by applying a horizontal flip and a resize-crop operation with a certain probability. Before training, images were also normalized in regards to the mean and the standard deviation of the Tiny ImageNet dataset.

The model was fine-tuned during 21 epochs, iterating over the triplets described in section 2.1.4 once per epoch. After each epoch, the triplet loss and accuracy were calculated both on training images and validation images. It took approximately 12 hours to train the network for 21 epochs. The model accepts parameters such as the regularization gap, learning rate, and l2-regularization coefficient. However, these hyperparameters were declared once and never optimized due to the learning process being very time-consuming. These values were, however, based on default values and optimal values found by Wang et al. [15].

2.1.6 Evaluation. Two evaluation metrics were used: accuracy; and a more subjective analysis of the image search results. Accuracy is defined as if the model classifies a triplet sample correctly. In other words, if the euclidean distance $D(.,.)$ between the query image p_i and the positive image p_i^+ than the distance between the positive image p_i and the negative image p_i^-. This means that the prediction is interpreted as correct if $D(p_i, p_i^+) < D(p_i, p_i^-)$. This definition of accuracy is also employed by Wang et al. [15] and thus considered suitable for the task at hand.

In order to evaluate the model visually, a more subjective approach was also taken. Given a query image, randomly sampled from the validation set, the five most similar images were identified. This was done by propagating all training images through the DRML to generate a space embedding of all training images. Randomly chosen query embeddings were retrieved in the same

[1]http://www.image-net.org/challenges/LSVRC/

manner. The five closest neighbors were then identified with respect to the euclidean distance of the embeddings. The analysis of the five most similar images to a query image can give an understanding of what the model classifies as similar. Furthermore, the 5 most divergent images to the query image were identified by finding the images with the largest euclidean distance from the query image.

2.2 Xception Model

Currently, image processing and image-related problems such as object detection, image classification, etc. are solved using CNNs [12]. The layers present in the CNN learns or extracts features such as pixels and shapes of objects as well as other characteristics of images which help solve various problems [10]. Once pre-trained models such as VGG16, Xception are applied on CNNs, the feature extraction is improved, increasing the performance of the model. After the introduction of deep learning network architectures such as LSTM (Long Short Term Memory), Bi-LSTM (Bidirectional Long Short Term Memory), etc; Natural Language Processing (NLP) related tasks are becoming easier to solve [9]. The combination of image and NLP techniques has led to solving many complex tasks such as emotion detection and text-generation from sign language for physically challenged people. This paper proposes a model solving the *Image Diversity* task by combining both CNN and LSTM architectures.

Image diversity, i.e finding images that are similar to the given input image can be solved in numerous ways, such as using Deep Ranking [15] or SimNet [1], etc. Another way of solving this task is to perform NLP tasks such as generating captions for a given image and calculating the text-similarity between the generated caption and the captions which are already available in the system (database, cloud, etc). The suggested model requires a lot of system resources and excessive time for training. To overcome this bottleneck, a transfer learning approach has been implemented, using, **Xception** [2], which is a pre-trained model applied on a CNN. As a result, this implementation achieves faster training times, improved model predictions, usage of fewer system resources, etc. The LSTM architecture is mainly used for sequence prediction problems such as text-generation [8]. This architecture is used for generating captions by using the details and information from the CNN architecture.

2.2.1 Model architecture.

The architecture, illustrated in Figure 2, is a simplified version based on the CNN-LSTM architecture proposed by Moses Soh [13]. Using this simplified model on images for caption generation is a time-consuming process. Hence, a pre-trained model is added before the final layer in the architecture to extract the features of given images. These features are saved to a file that can be loaded and used later. As a result, training times can be significantly reduced.

Simultaneously, the text data for each image undergoes preprocessing steps such as text to lower, removing stop words, removing symbols and numbers, and adding a *start* and *end* to each sentence. The reason behind adding a start and end tag for each sentence is that the start tag is a signal for the LSTM architecture to start predicting the next word based on the previous word and the end tag is to end the sequence prediction. These encoded sentences are then saved to a file for further process. Figure 3 presents the deep learning architecture with input and output size.

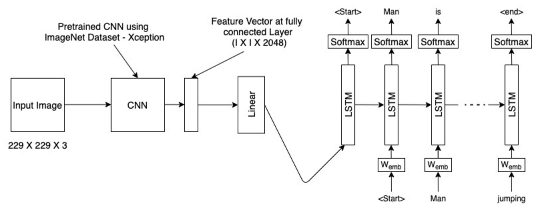


Figure 2: Image Caption Generator

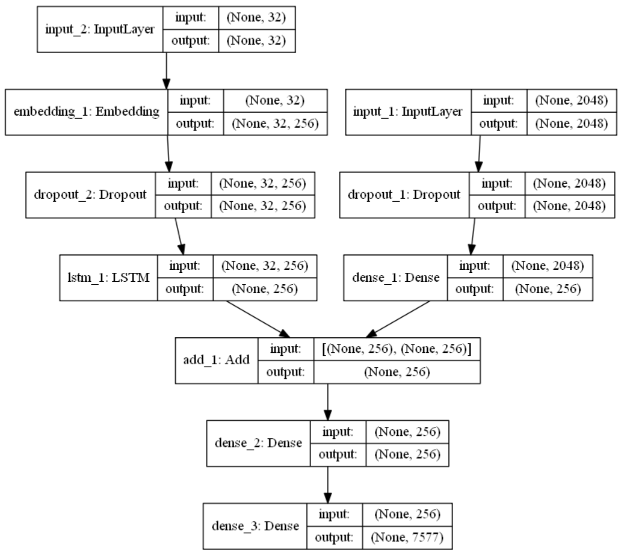

input_2: InputLayer | input: (None, 32) | output: (None, 32)
embedding_1: Embedding | input: (None, 32) | output: (None, 32, 256)
input_1: InputLayer | input: (None, 2048) | output: (None, 2048)
dropout_2: Dropout | input: (None, 32, 256) | output: (None, 32, 256)
dropout_1: Dropout | input: (None, 2048) | output: (None, 2048)
lstm_1: LSTM | input: (None, 32, 256) | output: (None, 256)
dense_1: Dense | input: (None, 2048) | output: (None, 256)
add_1: Add | input: [(None, 256), (None, 256)] | output: (None, 256)
dense_2: Dense | input: (None, 256) | output: (None, 256)
dense_3: Dense | input: (None, 256) | output: (None, 7577)

Figure 3: CNN-LSTM Deep Learning Architecture

The architecture consists of three major parts,

(1) Image feature extractor - The features extracted from images, of 2048 dimensions, are passed into the input layer of size 2048, then the output of the input layer is passed into the dropout layer to drop randomly chosen neurons to avoid overfitting. Now, regularized data is passed to the dense layer where the output is

$$output = activation(dot(input, kernel) + bias) \qquad (2)$$

. The dimension of the final output is squeezed to 256. The final layer in the actual CNN, i.e. classification layer, is removed, as the task at hand is not classification.

(2) Embedding - LSTM layer - Textual input is passed into the embedding layer of input size 32 (maximum size of all textual input). The embedding layer will use the index of the input texts and find the corresponding vectors from the Table of vectors. The output of the embedding layer is sent to the dropout layer for regularization. Finally, the regularized output is passed into the LSTM layer for sequence prediction.

(3) Final Layer - Merging Image feature extractor and Embedding-LSTM layer leads to the final prediction. This final layer is equal to the size of the vocabulary.

2.2.2 *Text-similarity.*

Text-similarity is a way to determine how closely two sentences are related to each other in terms of the lexical and semantic relationships. There are numerous ways to calculate text-similarity such as Cosine similarity, Euclidean distance, etc. In general, Cosine similarity is widely used for all similarity related tasks [16]. The formula for Cosine Similarity is illustrated in Figure 4.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

Figure 4: Cosine Similarity

The reason behind using Cosine Similarity is that at θ = 0, the cosine value is 1 and at θ = 180, the cosine value is -1. That is, when two vectors overlap with each other the cosine value will be higher and when two vectors are exactly opposite, the cosine value will be lower. The distance between vectors using cosine is measured as a 1-cosine value.

Similar images are fetched from the system (Database, cloud, etc) based on the text-similarity score between the generated caption for an input image using the CNN-LSTM model and a list of all captions available in the system. The threshold for text-similarity score is >= 0.90. That is, if the score between two sentences is greater than or equal to 0.90, then the corresponding image file is fetched and displayed to the user.

2.2.3 *Training.*

The network was trained using Keras [7] and the implementation was designed to run on both CPU and GPU. However, memory problems may occur while handling image loading on computers without a dedicated GPU. Hence, the model was trained on a *OMEN 1 7T-CB100* laptop.

Data plays a vital role in the success of deep learning models. Hence, variations such as lower/higher size of datasets, imbalanced classes, etc. will have impacts on the results [14]. Hence, both image data and text data are pre-processed before feeding into the model. The following processes are applied to images:

(1) Images are resized to 299 X 299.
(2) Images are normalized to bring the pixel values in the range of -1.0 to 1.0.

Once the images are pre-processed, the feature vector of the pre-processed images is extracted by using the Xception model and saved into a file for later use. The feature vectors are saved to a file to remove the need to run the Xception model on these pre-processed images again. Next, text data (captions) are read from the file which is separated by the next line (\n) character, which is then further split based on tab character(\t). The following processes are applied to the text data, that is, the captions.

(1) Captions are converted to lower case.
(2) Punctuations are removed from captions.
(3) Captions whose length greater than 1 are retained.

image vector	text sequence	Predicted sequence
feature vector	start	Man
feature vector	start,Man	is
feature vector	start,Man,is	jumping
feature vector	start,Man,is,jumping	here
feature vector	start,Man,is,jumping,here	end

Table 1: Structure of input and output data

(4) Numbers from captions are removed.

A vocabulary is created using the final output. The vocabulary and cleaned text along with the image names are saved into each files for later use. The structure of input and output data is illustrated in Table 1.

In Table 1, *image vector* and *text sequence* are inputs and *Predicted sequence* is output. The input is passed to the proposed model for 20 epochs. During the first and second epoch, there was a significant change in accuracy and loss on both training images and validation images. It took approximately 7.5 hours to train the network for 20 epochs. The model's hyperparameters were set to default values and were not optimized due to time constraints. The default values of the hyperparameters were the same as used in the Xception model [2].

2.2.4 *Evaluation.*

The deep learning model, that is, CNN-LSTM is evaluated on three metrics: accuracy, validation loss, and Bilingual Evaluation Understudy (BLEU); which is a comprehensive metric to test caption or text-generation models [17]. The accuracy is defined as if the model predicts captions correctly, that is, the number of correctly predicted captions divided by the total number of captions. To evaluate the performance of the model, the model is tested with validation data not previously seen by the model. The validation loss is another metric in the field of Machine Learning and Deep Learning to evaluate a model's performance. That is, the lower the validation loss, the better the model [5]. Finally, BLEU is a score obtained by comparing a generated caption to one or more reference captions or actual captions, as said in Xception pre-trained model [2].

In order to further analyze the performance of the model, a more practical approach was also taken. A query image, randomly selected from the given validation image dataset was chosen. This image, after passing through necessary pre-processing steps, was given as an input image to the trained model. The trained model generated the caption in less than 6-7 seconds. Once the caption was generated, it was compared with the actual caption for the particular input image and the text-similarity was calculated. The validation data went trough this process, resulting in a score above 0.80. Text-similarity was later calculated between the generated caption and all other captions available in the system and selected images whose text-similarity scores were above or equal to 0.90. This score yielded a list of similar images for the given query image along with their captions, Figure 14. In order to yield a list of dissimilar images, this text-similarity threshold was reduced from 0.90 to a value between 0.50 and 0.60. These scores yielded a list of divergent images along with their captions, presented in Figure 15.

3 RESULTS

3.1 Deep ranking model

Accuracy and loss for the training and validation set are presented in Figures 5, 6, 7 and 8. The training and validation accuracy are quite high, reaching more than 99%, and slightly less than 99% respectively. The loss decreases gradually without any major spikes. Note that Figures have varying scale on the y-axis and no smoothing has been applied.

The 5 most similar, and 5 most divergent images are presented in Figures 9 and 10 respectively. When observing the 5 most similar images, the model seems to have found some similarities, such as color and shape. The 5 most divergent images are quite unlike the query image, for example, a yellow bus is surely not similar to a fish. However, we consider it harder to determine what is 'divergent' as it might as well be random. Therefore, these results are harder to draw any conclusion from compared to the results displaying similar images.

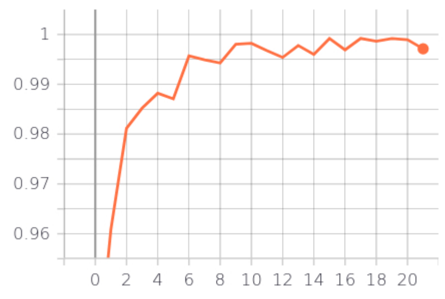


Figure 5: Training accuracy for 21 epochs (DRML).

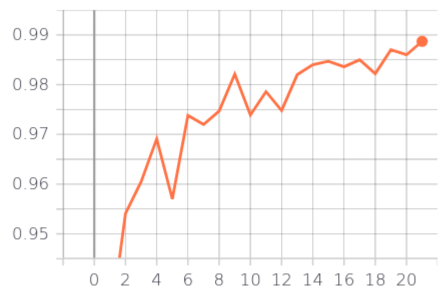


Figure 6: Validation accuracy for 21 epochs (DRML).

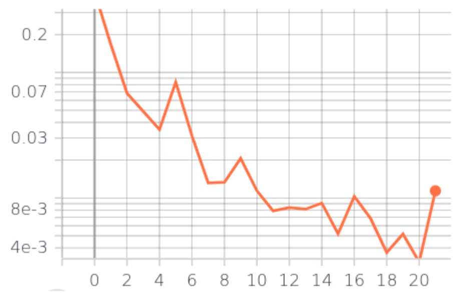


Figure 7: Training loss during 21 epochs (DRML).

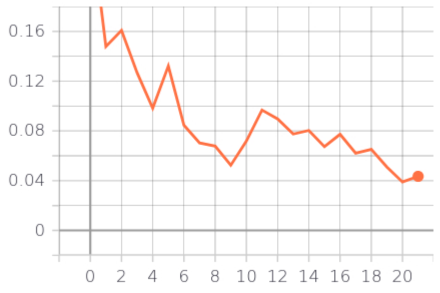


Figure 8: Validation loss during 21 epochs (DRML).

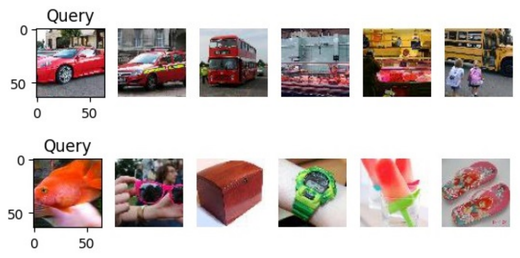


Figure 9: The 5 most similar images to a car and fish (DRML).

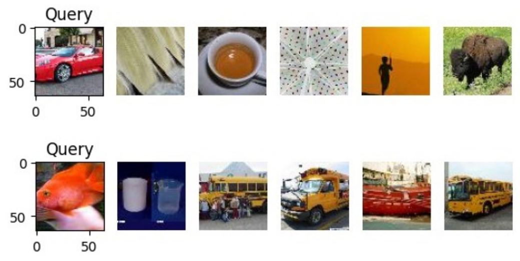


Figure 10: The 5 most divergent images to a car and fish (DRML).

3.2 Xception Model

Accuracy and loss for the training and validation set are presented in Figures 11, and 12. The training and validation accuracy is around 33% at epoch 4. The training and validation loss are 3.3206 and 3.6385 after 4 epochs. After 4 epochs, the validation and training loss did not significantly decrease any more. Hence, the model trained for 4 epochs was chosen as the final model.

The query image for which a caption is generated using the CNN-LSTM model, and the 10 most similar images are presented in Figures 13 and 14 respectively. When observing the 10 most similar images, 4 images are irrelevant to the given query image. From this, we can observe that out of 10 images, 6 similar images are fetched. That is, roughly 60% of the predictions were correct. The reason behind the poor performance of the model is explained in section 4.2.

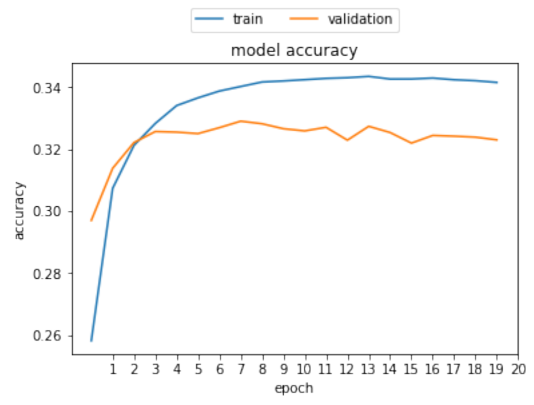


Figure 11: Training and Validation accuracy for 20 epochs (CNN-LSTM).

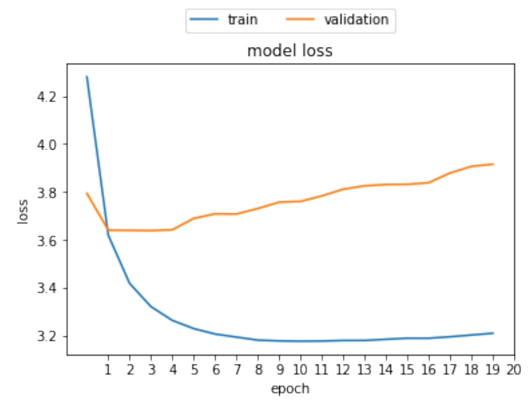
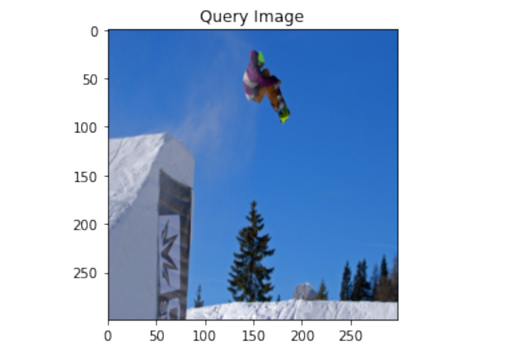
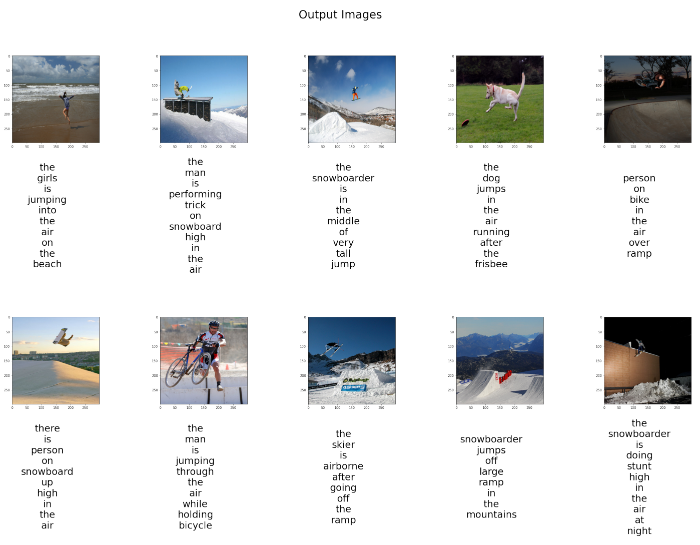


Figure 12: Training and Validation Loss for 20 epochs (CNN-LSTM).



snowboarder is jumping over the ramp in the air

Figure 13: Query image with CNN-LSTM model generated caption.



the girls is jumping into the air on the beach

the man is performing trick on snowboard high in the air

the snowboarder is in the middle of very tall jump

the dog jumps in the air running after the frisbee

person on bike in the air over ramp

there is person on snowboard up high in the air

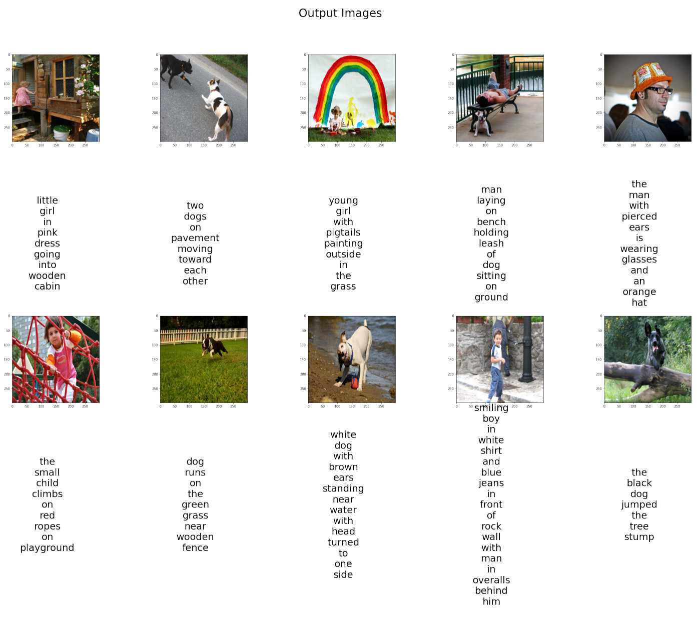
the man is jumping through the air while holding bicycle

the skier is airborne after going off the ramp

snowboarder jumps off large ramp in the mountains

the snowboarder is doing stunt high in the air at night

Figure 14: Similar images with captions using text-similarity.



little girl in pink dress going into wooden cabin

two dogs on pavement moving toward each other

young girl with pigtails painting outside in the grass

man laying on bench holding leash of dog sitting on ground

the man with pierced ears is wearing glasses and an orange hat

the small child climbs on red ropes on playground

dog runs on the green grass near wooden fence

white dog with brown ears standing near water with head turned to one side

smiling boy in white shirt and blue jeans in front of rock wall with man in overalls behind him

the black dog jumped the tree stump

Figure 15: Divergent images with captions using text-similarity.

4 CONCLUSIONS AND DISCUSSION

4.1 Deep ranking model

The training and validation accuracy are quite high, reaching approximately 99.5% for the training set, and slightly less than 99% for the validation set. Initially, the high performance was a bit surprising. However, the ResNet-18 has been trained to classify more than 1000 classes, thus it is reasonable for the DRML to have quite good accuracy as well, especially due to the choice of in- and out-of-class images for the triplets. Furthermore, a downward trend of the loss and training can be observed. Thus, training for a longer period of time would perhaps result in even better accuracy. During the last epoch, there is a spike in the training loss, see Figure 7. However, this spike has no significant meaning as the scale of the y-axis is quite small compared to the other plots.

Image similarity has no definite answer and thus is not an easy task. How similar two images are interpreted can vary from person to person. The DRML needs to decide the degree of similarity between images based on different patterns in the images. Only using the ResNet-18, image similarity would be considered on a category-level, where, two images would be considered similar if they belong to the same category. The goal of the DRM was to create a more fine-grained image similarity classifier that is able to recognize more fine-grained differences between images of the same class. The DRML seems to have found some similar traits in images, e.g. the color and shape. As seen in Figure 9, in the example with a car as query, other cars and buses are considered similar. However, a red car and a meat counter are also considered to be similar, which implies that the model has identified some fine-grained features such as color. This was also quite evident in the second example with a fish as a query image.

The choice of triplets impacts what the model considers as similar, and consequently also dissimilar. When generating triples, positive images were sampled from the same class as the query image, while negative images were sampled from the out-of-class images. Thus it would be reasonable to think that the choice of triplets would entail that images of the same class are similar. However, the results show signs of more fine-grained behavior. For example, using the fish query in Figure 9, it was initially believed that other fish would be considered similar. Nevertheless, it turned out that no other fish was considered to be similar.

In conclusion, using DRML to find divergent images can be a valid choice. However, one has to consider what *divergent* means, and take this into account when generating triplets. The implemented model is perhaps better suited for image similarity, as these results are more intuitive and provide a better understanding of the model's behavior than it does regarding image diversity. A possible approach to make the model better suited for image diversity could be to look at how similar new images are to the images the user has already seen or liked. Thus, also considering user history.

Furthermore, a multi-scale network architecture could be used instead of single scale CNN, as a single-scale CNN has very strong invariance encoded in the model, which may hinder fine-grained image similarity recognition [15]. Another improvement would be to include of in-class negative samples. Intuitively, such an improvement will increase the model's capability to recognize differences between images of the same class. This is further supported by Wang et al. [15], who discovered that the score-at-top-30 increased as they increased the proportion in-class negative samples increased. However, they found having at least a fraction, about 20%, of out of class negative samples improved the accuracy a lot, which favors the simplified triplet sampling implemented in this project. The score-at-top-30 is defined as the number of correctly ranked triplets minus the number of incorrectly ranked ones on the number of triplets whose positive image or negative image is among the top 30 closest images of the query image [15].

4.2 Xception model

The training and validation accuracy of the final model is quite low, reaching around 33% for both training and validation data. The training and validation loss are 3.32 and 3.64. However, these results were expected as the model was trained on only 12,000 images. In order to attain better model predictions, the model most likely has to be trained using more than 50.000 images along with captions. However, this model does pretty well in predicting or generating captions for a given input image.

The Xception model was chosen over other pre-trained image models because, as per Francois Chollet [2], Xception models yield a good accuracy when used on image classification tasks. Hence, features extracted from images would be used to get good predictions when applied on a CNN architecture for image-related tasks such as image classification. Moreover, applying text-similarity techniques paved a new approach of fetching similar images and divergent images based on the similarity between generated captions, rather than relying on image pixels comparisons.

In conclusion, Using CNN-LSTM along with text-similarity to find divergent images is a new approach that joins Image Processing and NLP together. The model trained is very well suited to predict both similar images and divergent images by altering the threshold for text similarity. But, before using this model, one should carefully consider what makes two images similar or divergent to each other. The results produced by this model may contain some divergent images, in case of similar image search and vice versa. To avoid this, the feature comparison between the given image and the set of images can be implemented to remove any irrelevant images.

4.2.1 *Further improvements.*

This approach could be further improved by performing any one of the following items,

(1) Applying various pre-trained image models such as VGG16, ResNet-152, Inception V3, etc.
(2) Using COCO dataset which consists of around 100,000 images to train the model.
(3) Fine tuning the model's hyperparameters.
(4) Using pre-trained word embeddings for text generation or using Transformers; a new Deep Learning approach for sequence generation.

REFERENCES

[1] Srikar Appalaraju and Vineet Chaoji. 2017. Image similarity using deep CNN and curriculum learning. *arXiv preprint arXiv:1709.08761* (2017).

[2] François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 1251–1258.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]* (Dec. 2015). http://arxiv.org/abs/1512.03385 arXiv: 1512.03385.

[4] ImageNet. 2020. About ImageNet. Retrieved November 25, 2020 from http://image-net.org/about-overview

[5] Jacob Kasche and Fredrik Nordström. 2020. Regularization Methods in Neural Networks.

[6] Keggle. 2016. Tiny ImageNet. Retrieved November 25, 2020 from https://www.kaggle.com/c/tiny-imagenet/overview

[7] Keras. 2020. Keras. Retrieved December 13, 2020 from https://keras.io/

[8] Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text generation by structure-aware seq2seq learning. *arXiv preprint arXiv:1711.09724* (2017).

[9] Zhengdong Lu and Hang Li. 2016. Recent progress in deep learning for NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*. 11–13.

[10] Joe Yue-Hei Ng, Fan Yang, and Larry S. Davis. 2015. Exploiting local features from deep networks for image retrieval. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, Boston, MA, USA, 53–61. https://doi.org/10.1109/CVPRW.2015.7301272

[11] Pytorch. 2020. Pytorch. Retrieved November 26, 2020 from https://pytorch.org

[12] H. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. 2016. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging* 35, 5 (May 2016), 1285–1298. https://doi.org/10.1109/TMI.2016.2528162 Conference Name: IEEE Transactions on Medical Imaging.

[13] Moses Soh. 2016. Learning CNN-LSTM architectures for image caption generation. *Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep* (2016).

[14] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*. 843–852.

[15] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. 2014. Learning Fine-Grained Image Similarity with Deep Ranking. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 1386–1393. https://doi.org/10.1109/CVPR.2014.180

[16] Peipei Xia, Li Zhang, and Fanzhang Li. 2015. Learning similarity with cosine similarity ensemble. *Information Sciences* 307 (2015), 39–52.

[17] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1097–1100.