Deep Visual-Semantic Alignments for Generating Image Descriptions

Introduction

• Goal: Generating dense descriptions of images

• Challenges:
  • Design of a model that is rich enough to simultaneously reason about contents of images and their representation in the domain of natural language
  • Datasets of image captions are available, but these descriptions multiplex mentions of several entities whose locations in images are unknown
Introduction

• Core insight: We can leverage these large image-sentence datasets by treating the sentences as weak labels

• Contributions
  • We develop a deep neural network model that infers the latent alignment between segments of sentences and the region of the image that they describe
  • We introduce a multimodal recurrent neural network that takes an input image and generates its description in text

• Code is publicly available
  • http://cs.stanford.edu/people/karpathy/deepimagesent
Model

• During training, the input is a set of images and their corresponding sentence descriptions

• 1. Align sentence snippets to visual regions

• 2. Treat these correspondences as training data for a multimodal RNN model that learns to generate the snippets
Learning to Align Visual and Language Data

• Representing images
  • Detect objects in every image with a Region Convolutional Neural Network (RCNN).
  • The CNN is pre-trained on ImageNet and finetuned on the 200 classes of the ImageNet Detection Challenge.

Learning to Align Visual and Language Data

• Representing images
  • We use the top 19 detected locations in addition to the whole image and compute the representations based on the pixels $I_b$ inside each bounding box
    $$v = Wm[CNN_{\theta_c}(I_b)] + bm$$
  • Where $CNN(Ib)$ transforms the pixels inside bounding box $I_b$ into 4096-dimensional activations of the fully connected layer immediately before the classifier. The CNN parameters $\theta_c$ contain approximately 60 million parameters. The matrix $W_m$ has dimension $h \times 4096$, where $h$ is the size of the multimodal embedding space.
  • Every image is thus represented as a set of $h$-dimensional vectors $\{v_i|i = 1, \ldots, 20\}$
Learning to Align Visual and Language Data

• Representing sentences
  • We would like to represent the words in the sentence in the same $h$-dimensional embedding space that the image regions occupy.
  • We propose to use Bidirectional Recurrent Neural Network (BRNN) to compute word representations.
  • The BRNN takes a sequence of $N$ words (encoded in a 1-of-$k$ representation) and transforms each one into an $h$-dimensional vector.
  • However, the representation of each word is enriched by a variably-sized context around the word.
Learning to Align Visual and Language Data

• Representing sentences
  • Using the index $t = 1, \ldots, N$ to denote the position of a word in a sentence, the precise form of BRNN is as follows.

$$x_t = W_w \mathbb{I}_t$$
$$e_t = f(W_e x_t + b_e)$$
$$h_t^f = f(e_t + W_f h_{t-1}^f + b_f)$$
$$h_t^b = f(e_t + W_b h_{t+1}^b + b_b)$$
$$s_t = f(W_d (h_t^f + h_t^b) + b_d)$$

• Here, $\mathbb{I}_t$ is an indicator column vector that has a single one at the index of the $t$-th word in a word vocabulary. The weights $W_w$ specify a word embedding matrix that we initialize with 300-dimensional word2vec weights and keep fixed due to overfitting concerns.
Learning to Align Visual and Language Data

- Representing sentences
  - The final $h$-dimensional representation $s_t$ for the $t$-th word is a function of both the word at that location and also its surrounding context in the sentence.
  - We learn the parameters $W_e, W_f, W_b, W_d$ and the representation biases $b_e, b_f, b_b, b_d$.
  - A typical size of the hidden representation ranges between 300-600 dimensions.
  - We set the activation function $f$ to the rectified linear unit (ReLU), which computes $f: x \rightarrow \max(0, x)$
Learning to Align Visual and Language Data

• Alignment objective
  • Formulate an image-sentence score as a function of the individual region-word scores.
  • The model of Karpathy et al. interprets the dot product $v_i^T s_t$ between the $i$-th region and $t$-th word as a measure of similarity and use it to define the score between image $k$ and sentence $l$ as:
    $$ S_{kl} = \sum_{t \in g_l} \sum_{i \in g_k} \max(0, v_i^T s_t) $$
  • Here, $g_k$ is the set of image fragments in image $k$ and $g_l$ is the set of sentence fragments in sentence $l$.

Learning to Align Visual and Language Data

• Alignment objective
  • The following reformulation simplifies the model
    \[ S_{kl} = \sum_{t \in g_l} \max_{i \in g_k} v_i^T s_t \]
  • Here, every word \( s_t \) aligns to the single best image region. This simplified model leads to improvements in the final ranking performance.
Learning to Align Visual and Language Data

• Decoding text segment alignments to images
  • We can interpret the quantity $v_i^T s_t$ as the un-normalized log probability of the $t$-th word describing any of the bounding boxes in the image.
  • However, since we are interested in generating snippets of text instead of single words, we would like to align extended, contiguous sequences of words to a single bounding box.
  • We treat the true alignment as latent variables in a Markov Random Field (MRF) where the binary interactions between neighboring words encourage an alignment to the same region.
Learning to Align Visual and Language Data

• Decoding text segment alignments to images
  • Given a sentence with $N$ words and an image with $M$ bounding boxes, we introduce the latent alignment variables $a_j \in \{1, \ldots, M\}$ for $j = 1, \ldots, N$ and formulate an MRF in a chain structure along the sentence as follows
  
  \[ E(\alpha) = \sum_{j=1,\ldots,N} \psi^U_j(a_j) + \sum_{j=1,\ldots,N-1} \psi^B_j(a_j, a_{j+1}) \]

  \[ \psi^U_j(a_j = t) = v^T_i s_t \]

  \[ \psi^B_j(a_j, a_{j+1}) = \beta \mathbb{1}[a_j = a_{j+1}] \]

  • Here, $\beta$ is a hyperparameter that controls the affinity towards longer word phrases.
  • We minimize the energy to find the best alignments $\alpha$ using dynamic programming. The output is a set of image regions annotated with segments of text.
Multimodal RNN for Generating Descriptions

• Key challenge: designing a model that can predict a variable-sized sequence of outputs given an image

• During training the Multimodal RNN takes the image pixels $I$ and a sequence of input vectors $(x_1, ..., x_T)$. It then computes a sequence of hidden states $(h_1, ..., h_t)$ and a sequence of outputs $(y_1, ..., y_t)$ by iterating the following recurrence relation for $t = 1$ to $T$:

$$b_v = W_{hi}[\text{CNN}_{\theta_c}(I)]$$

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + \mathbb{1}(t = 1) \otimes b_v)$$

$$y_t = \text{softmax}(W_{oh}h_t + b_o)$$
Multimodal RNN for Generating Descriptions

• The output vector $y_t$ holds the (unnormalized) log probabilities of words in the dictionary and one additional dimension for a special END token.

• We provide the image context vector $b_v$ to the RNN only at the first iteration, which we found to work better than at each time step.

• In practice we also found that it can help to also pass both $b_v, W_{hx} x_t$ through the activation function.

• A typical size of the hidden layer of the RNN is 512 neurons.
Multimodal RNN for Generating Description

• RNN Training
  • The RNN is trained to combine a word $x_t$, the previous context $h_{t-1}$ to predict the next word $y_t$. We condition the RNN’s predictions on the image information $b_v$ via bias interactions on the first step.
  • We set $h_0 = \mathbf{0}$, $x_1$ to a special START vector, and the desired label $y_1$ as the first word in the sequence.
  • We set $x_2$ to the word vector of the first word and expect the network to predict the second word, etc.
  • $x_T$ corresponds to a special END token
  • The cost function is to maximize the log prob. assigned to the target labels (i.e., Softmax classifier).
Multimodal RNN for Generating Descriptions

- RNN at test time
  - To predict a sentence, we compute the image representation $b_v$, set $h_0 = 0$, $x_1$ to the START vector and compute the distribution over the first word $y_1$.
  - We sample a word from the distribution (or pick the argmax), set its embedding vector as $x_2$, and repeat this process until the END token is generated.
Optimization

• We use SGD with mini-batches of 100 image-sentence pairs and momentum of 0.9 to optimize the alignment model.

• We also use dropout regularization in all layers except in the recurrent layers.

• We achieved the best results using RMSprop, which is an adaptive step size method that scales the update of each weight by a running average of its gradient norm.
Experiments

• Datasets
  • Flickr8K: 8,000 images
  • Flickr30K: 31,000 images
  • MSCOCO: 123,000 images
  • Each image is annotated with 5 sentences using Amazon Mechanical Turk
  • For Flickr8K and Flickr30K, we use 1,000 images for validation, 1,000 images for testing, and the rest for training
  • For MSCOCO we use 5,000 images for both validation and testing
Experiments

• Data Preprocessing
  • We convert all sentences to lower-case, discard non-alphanumeric characters
  • We filter words to those that occur at least 5 times in the training set, which results in 2538, 7414, and 8791 words for Flickr8K, Flickr30K, and MSCOCO datasets, respectively.
Image-Sentence Alignment Evaluation

• Retrieve items in one modality given a query from the other by sorting based on the image-sentence score $S_{kl}$. We report the median rank of the closest ground truth result in the list and Recall@K.

<table>
<thead>
<tr>
<th>Model</th>
<th>Image Annotation</th>
<th></th>
<th></th>
<th>Image Search</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@1</td>
<td>R@5</td>
<td>R@10</td>
<td>Med r</td>
<td>R@1</td>
<td>R@5</td>
</tr>
<tr>
<td>SDT-RNN (Socher et al. [49])</td>
<td>9.6</td>
<td>29.8</td>
<td>41.1</td>
<td>16</td>
<td>8.9</td>
<td>29.8</td>
</tr>
<tr>
<td>Kiros et al. [25]</td>
<td>14.8</td>
<td>39.2</td>
<td>50.9</td>
<td>10</td>
<td>11.8</td>
<td>34.0</td>
</tr>
<tr>
<td>Mao et al. [38]</td>
<td>18.4</td>
<td>40.2</td>
<td>50.9</td>
<td>10</td>
<td>12.6</td>
<td>31.2</td>
</tr>
<tr>
<td>Donahue et al. [8]</td>
<td>17.5</td>
<td>40.3</td>
<td>50.8</td>
<td>9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DeFrag (Karpathy et al. [24])</td>
<td>14.2</td>
<td>37.7</td>
<td>51.3</td>
<td>10</td>
<td>10.2</td>
<td>30.8</td>
</tr>
<tr>
<td>Our implementation of DeFrag [24]</td>
<td>19.2</td>
<td>44.5</td>
<td>58.0</td>
<td>6.0</td>
<td>12.9</td>
<td>35.4</td>
</tr>
<tr>
<td>Our model: DepTree edges</td>
<td>20.0</td>
<td>46.6</td>
<td>59.4</td>
<td>5.4</td>
<td>15.0</td>
<td>36.5</td>
</tr>
<tr>
<td>Our model: BRNN</td>
<td>22.2</td>
<td>48.2</td>
<td>61.4</td>
<td>4.8</td>
<td>15.2</td>
<td>37.7</td>
</tr>
<tr>
<td>Vinyals et al. [54] (more powerful CNN)</td>
<td>23</td>
<td>-</td>
<td>63</td>
<td>5</td>
<td>17</td>
<td>-</td>
</tr>
<tr>
<td>MSCOCO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our model: 1K test images</td>
<td>38.4</td>
<td>69.9</td>
<td>80.5</td>
<td>1.0</td>
<td>27.4</td>
<td>60.2</td>
</tr>
<tr>
<td>Our model: 5K test images</td>
<td>16.5</td>
<td>39.2</td>
<td>52.0</td>
<td>9.0</td>
<td>10.7</td>
<td>29.6</td>
</tr>
</tbody>
</table>
Image-Sentence Alignment Evaluation

- Our simpler cost function improves performance
  - Comparing with DeFrag, the only difference is the new, simpler cost function
- BRNN outperforms dependency tree relations
- MSCOCO results for future comparisons
- Learned region and word vector magnitudes
  - An appealing feature of our model is that it learns to modulate the magnitude of the region and word embeddings
  - Visually discriminative words have embedding vectors with higher magnitudes
Image-Sentence Alignment Evaluation

- Results before MRF smoothing
- Small or relatively rare objects can be described, e.g., accordion
Generated Descriptions: Fullframe Evaluation

• We first train our Multimodal RNN to generate sentences on full images. We use the more powerful VGGNet image features.

• We report the BLEU, METEOR, and CIDEr scores

Generated Descriptions: Fullframe Evaluation

- Nearest neighbor retrieval baseline: annotate each test image with a sentence of the most similar training set image
- Multimodal RNN confidently outperforms this retrieval method
- [54] uses a more powerful but more complex sequence learner (LSTM) and a different CNN.

<table>
<thead>
<tr>
<th>Model</th>
<th>Flickr8K</th>
<th>Flickr30K</th>
<th>MSCOCO 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B-1</td>
<td>B-2</td>
<td>B-3 B-4</td>
</tr>
<tr>
<td>Nearest Neighbor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mao et al. [38]</td>
<td>58</td>
<td>28</td>
<td>23</td>
</tr>
<tr>
<td>Google NIC [54]</td>
<td>63</td>
<td>41</td>
<td>27</td>
</tr>
<tr>
<td>LRCN [8]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS Research [12]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our model</td>
<td>57.9</td>
<td>38.3</td>
<td>24.5</td>
</tr>
</tbody>
</table>

Table 2. Evaluation of full image predictions on 1,000 test images. **B-n** is BLEU score that uses up to n-grams. High is good in all columns. For future comparisons, our METEOR/CIDEr Flickr8K scores are 16.7/31.8 and the Flickr30K scores are 15.3/24.7.
Generated Descriptions: Region Evaluation

- Use AMT to collect a new dataset. We collected 9,000 text snippets for 200 images in our MSCOCO test split. The snippets have an average length of 2.3 words.
- “table with wine glasses” only occurs in the training set 30 times. Each time it may have a different appearance.
- The model had to first correctly learn to ground the string and then also learn to generate it.
Generated Descriptions: Region Evaluation

- Region model outperforms full frame model and ranking baseline
  - The ranking baseline retrieves training sentence substrings most compatible with each region as judged by the BRNN model.
  - The region model also outperforms the full frame model on all other metrics: CIDEr 61.6/20.3, METEOR 15.8/13.3, ROUGE 35.1/21.0 for region/fullframe respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>B-1</th>
<th>B-2</th>
<th>B-3</th>
<th>B-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human agreement</td>
<td>61.5</td>
<td>45.2</td>
<td>30.1</td>
<td>22.0</td>
</tr>
<tr>
<td>Nearest Neighbor</td>
<td>22.9</td>
<td>10.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>RNN: Fullframe model</td>
<td>14.2</td>
<td>6.0</td>
<td>2.2</td>
<td>0.0</td>
</tr>
<tr>
<td>RNN: Region level model</td>
<td>35.2</td>
<td>23.0</td>
<td>16.1</td>
<td>14.8</td>
</tr>
</tbody>
</table>

Table 3. BLEU score evaluation of image region annotations.
Limitations

• The model can only generate a description of one input array of pixels at a fixed resolution.

• The RNN receives the image information only through additive bias interactions, which are known to be less expressive than more complicated multiplicative interactions.

• Our approach consists of two separate models. Going directly from an image-sentence dataset to region-level annotations as part of a single model trained end-to-end remains an open problem.
Conclusions

• We introduce a model that generates natural language description of image regions

• We show that this model provides state of the art performance on image-sentence ranking experiments

• We describe a multimodal RNN architecture that generates descriptions of visual data
Show and Tell: A Neural Image Caption Generator

Introduction

• Goal: A single joint model that takes an image $I$ as input, and is trained to maximize the likelihood $p(S|I)$ of producing a target sequence of words $S = \{S_1, S_2, \ldots\}$ where each word $S_t$ comes from a given dictionary

• Idea from machine translation: transform a sentence $S$ written in a source language into its translation $T$ in the target language, by maximizing $p(T|S)$
Introduction

• Contributions
  • Present a neural net that is fully trainable using stochastic gradient descent
  • This model combines state-of-art sub-networks for vision and language models
  • It yields significantly better performance compared to state-of-the-art approaches
Model

• Recent statistical machine translation: Making use of a recurrent neural network which encodes the variable length input into a fixed dimensional vector, and uses this representation to “decode” it to the desired output sentence.

• Maximize the probability of the correct description given the image

\[
\theta^* = \arg\max_{\theta} \sum_{I,S} \log p(s|I; \theta)
\]

• Where \( \theta \) are the parameters of our model, \( I \) is an image, and \( S \) its correct transcription.
Model

• It is common to apply the chain rule to model the joint probability over \( S_0, ..., S_N \) where \( N \) is the length of this particular example as

\[
\log p(S|I) = \sum_{t=0}^{N} \log p(S_t|I, S_0, ..., S_{t-1})
\]

• At training time, \((S, I)\) is a training example pair, and we optimize the sum of the log probabilities over the whole training set using stochastic gradient descent.
Model

• Model $p(S_t | I, S_0, \ldots, S_{t-1})$ with a Recurrent Neural Network (RNN)
  • The variable number of words we condition upon up to $t - 1$ is expressed by a fixed length hidden state or memory $h_t$
  • This memory is updated after seeing a new input $x_t$ by using a non-linear function $f: h_{t+1} = f(h_t, x_t)$

• Two crucial design
  • What is the exact form of $f$ -- Long-Short Term Memory (LSTM)
  • How are the images and words fed as inputs $x_t$ -- Convolutional Neural Network (CNN) for image representation, embedding model for words
LSTM- Based Sentence Generator

• The core of LSTM is a memory cell \( c \) encoding knowledge at every time step of what inputs have been observed up to this step.

• The behavior of the cell is controlled by “gates” – layers which are applied multiplicatively and thus can either keep a value from the gated layer if the gate is 1 or zero this value if the gate is 0.
LSTM-Based Sentence Generator

• The definition of the gates and cell update and output

\[ i_t = \sigma(W_{ix} x_t + W_{im} m_{t-1}) \]
\[ f_t = \sigma(W_{fx} x_t + W_{fm} m_{t-1}) \]
\[ o_t = \sigma(W_{ox} x_t + W_{om} m_{t-1}) \]
\[ c_t = f_t \odot c_{t-1} + i_t \odot h(W_{cx} x_t + W_{cm} m_{t-1}) \]
\[ m_t = o_t \odot c_t \]
\[ p_{t+1} = \text{Softmax}(m_t) \]

• Where \( \odot \) represents the product with a gate value, and the various \( W \) matrices are trained parameters.

• The nonlinearities are sigmoid \( \sigma() \) and hyperbolic tangent \( h() \).

• The last equation \( m_t \) is what is used to feed to a Softmax, which will produce a probability distribution \( p_t \) over all words.
LSTM-Based Sentence Generator

• Training
  • The LSTM model is trained to predict each word of the sentence after it has seen the image as well as all preceding words as defined by $p(S_t|I,S_0,...,S_{t-1})$.
  • A copy of the LSTM memory is created for the image and each sentence word such that all LSTMs share the same parameters and the output $m_{t-1}$ of the LSTM at time $t-1$ is fed to the LSTM at time $t$.
LSTM-Based Sentence Generator

• Training
  • If we denote by $I$ the input image and by $S = (S_0, ..., S_N)$ a true sentence describing this image, the unrolling procedure reads:

  \[
  x_{-1} = \text{CNN}(I) \\
  x_t = W_e S_t, \quad t \in \{0 \ldots N - 1\} \\
  p_{t+1} = \text{LSTM}(x_t), \quad t \in \{0 \ldots N - 1\}
  \]

  • Where we represent each word as a one-hot vector $S_t$ of dimension equal to the size of the dictionary.

  • The image $I$ is only input once, at $t = -1$, to inform the LSTM about the image contents.
LSTM-Based Sentence Generator

• Training
  • Our loss is the sum of the negative log likelihood of the correct word at each step as follows
  
  \[
  L(I, S) = - \sum_{t=1}^{N} \log p_t(S_t)
  \]
  
  • The above loss is minimized w.r.t. all the parameters of the LSTM, the top layer of the image embedder CNN and word embeddings \(W_e\)
LSTM-Based Sentence Generator

• Inference
  • Sampling: We just sample the first word according to $p_1$, then provide the corresponding embedding as input and sample $p_2$, continuing like this until we sample the special end-of-sentence token or some maximum length
  • BeamSearch: Iteratively consider the set of the $k$ best sentences up to time $t$ as candidates to generate sentences of size $t + 1$, and keep only the resulting best $k$ of them. This better approximates
    \[ S = \text{argmax}_S p(S'|I) \]
  • We use the BeamSearch approach, with a beam of size 20
Experiments

• Datasets
  • With the exception of SBU, each image has been annotated by labelers with 5 sentences.
  • SBU consists of description given by image owners when they uploaded them to Flickr.
  • In the case of SBU, we hold out 1000 images for testing and train on the rest.
  • We reserve 4K random images from the MSCOCO validation set as test, and use it to report results.

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
</tr>
<tr>
<td>Pascal VOC 2008 [6]</td>
<td>-</td>
</tr>
<tr>
<td>Flickr8k [26]</td>
<td>6000</td>
</tr>
<tr>
<td>Flickr30k [33]</td>
<td>28000</td>
</tr>
<tr>
<td>MSCOCO [20]</td>
<td>82783</td>
</tr>
<tr>
<td>SBU [24]</td>
<td>1M</td>
</tr>
</tbody>
</table>
Experiments

• Training details
  • Preventing overfitting: Initialize the weights of the CNN component to a predefined model (e.g., on ImageNet)
  • Dropout and ensembling gave a few BLEU points improvement
  • We trained all sets of weights using stochastic gradient descent with fixed learning rate and no momentum.
  • We used 512 dimension for the embeddings and the size of the LSTM memory.
  • Keeping all words that appeared at least 5 times in the training set
Experiments

- Human scores were computed by comparing one of the human captions against the other four. We do this for each of the five raters, and average their BLEU scores.

- Study whether we could transfer a model to a different dataset. Comparing Flickr30k and Flickr8k, we see gains by adding more training data.
Experiments

• Generation diversity
  • Whether the generated captions are both diverse and high quality?
  • The agreement in BLEU score between the top 15 generated sentences is 58, which is similar to that of humans among them
  • In bold are the sentences that are not present in the training set
  • If we take the best candidate, the sentence is present in the training set 80% of the times
  • If we analyze the top 15 generated sentences, about half of the times we see a completely novel description
Experiments

- Ranking results
- Human evaluation
  - NIC is better than the reference system, but is clearly worse than the ground truth.
  - This shows that BLEU is not a perfect metric, as it does not capture well the difference between NIC and human descriptions.
Experiments
Conclusion

• NIC is based on a convolution neural network that encodes an image into a compact representation, followed by a recurrent neural network that generates a corresponding sentence.
• Work better than other systems on several datasets
• As the size of the datasets increase, so will the performance
DeepFont: Identify Your Font from An Image

Introduction

• Motivation
  • Typography is fundamental to graphic design
  • The majority of font selection interfaces are simple line lists

• Challenges
  • Huge space of possible fonts
  • Dynamic and open-ended properties of font classes
  • Very subtle and character-dependent difference among fonts
Introduction

• Contributions
  • Adobe VFR dataset: labeled and unlabeled real-world images
  • Domain adapted CNN: extend generality
  • Learning-based model compression
Dataset

- Domain mismatch between synthetic and real-world data
- AdobeVFR dataset
  - VFR_rea_test: 4,384 real-world test images with reliable labels, covering 617 classes
  - VFR_real_u: 197,396 unlabeled real-world images
  - VFR_sync_train and VFR_syn_val: render long English words sampled from a large corpus, and generate tightly cropped, grayscale, and size-normalized text images. For each class, 1,000 images for training, and 100 for validation.

Figure 2: (a) the different characters spacings between a pair of synthetic and real-world images. (b) the different aspect ratio between a pair of synthetic and real-world image
Dataset

• Synthetic data augmentation: Before feeding synthetic data into model training, augment training data using label-preserving transformations to reduce overfitting
  • 1. Noise
  • 2. Blur
  • 3. Perspective rotation
  • 4. Shading
  • 5. Variable character spacing
  • 6. Variable aspect ratio

Figure 3: The effects of data augmentation steps. (a)-(d): synthetic images of the same text but with different data augmentation ways. (e) compares relative differences of (a)-(d) with the real-world image Fig. 2 (a), in the measure of layer-wise network activations through the same DeepFont model.
Domain Adapted CNN

- CNN architecture further decomposed into two sub-networks
  - Unsupervised cross-domain sub-network $C_u$: will be trained using unlabeled data from both synthetic and real-world images
  - Supervised domain-specific sub-network $C_s$: learning higher-level discriminative features for classification, based on the shared features from $C_u$, using labeled data from the synthetic domain only
Domain Adapted CNN

• Training details
  • Train $C_u$ using mean squared error as the loss function
  • The dropout technique is applied to fc6 and fc7
  • Both $C_u$ and $C_s$ are trained with a default batch size of 128, momentum of 0.9 and weight decay of 0.0005

• Testing details
  • For each test image, it is first normalized to 105 pixels in height, but squeezed in width by three different random ratios
  • Under each squeezed scale, five $105 \times 105$ patches are sampled at different random locations
  • As every single patch could produce a softmax vector, we average all fifteen softmax vectors to determine the final classification result of the test image
Learning-Based Model Compression

• Deep models are heavily over-parameterized
• One way to shrink the number of parameters is using matrix factorization. Given the parameter $W \in \mathbb{R}^{m \times n}$, we use SVD

$$W = U S V^T$$

• Where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are two dense orthogonal matrices, and $S \in \mathbb{R}^{m \times n}$ is a diagonal matrix.
• To restore an approximate $W$, we can utilize $\bar{U}, \bar{V}, \bar{S}$, which denote the submatrices corresponding to the top $k$ singular vectors

$$\bar{W} = \bar{U} \bar{S} \bar{V}^T$$
Learning-Based Model Compression

• Instead of first training a model then lossy-compressing its parameters, we propose to directly learn a losslessly compressible model.

• Our goal is to make sure that the parameter matrix $W$’s rank is exactly no more than a small constant $k$.

• Hard thresholding operation is executed on $W$ after it is updated

$$W_k = UT_k(S)V^T$$

• Where $T_k$ will keep the largest $k$ eigenvalues in $S$
Experiments

• Analysis of domain mismatch
  • Sharp performance drop from N to R of SCAE N indicates that the convolutional features for synthetic and real data are quite different
  • This gap is reduced in SCAE S, and further in SCAE F
  • SCAE R fits the real-world data best
  • SCAE FR achieves an overall best reconstruction performance

<table>
<thead>
<tr>
<th>Methods</th>
<th>Training Data</th>
<th>Train N</th>
<th>Train R</th>
<th>Test N</th>
<th>Test R</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCAE N</td>
<td>N: VFR_syn_train, no data augmentation</td>
<td>0.02</td>
<td>3.54</td>
<td>31.28</td>
<td></td>
</tr>
<tr>
<td>SCAE S</td>
<td>S: VFR_syn_train, standard augmentation 1-4</td>
<td>0.21</td>
<td>2.24</td>
<td>19.34</td>
<td></td>
</tr>
<tr>
<td>SCAE F</td>
<td>F: VFR_syn_train, full augmentation 1-6</td>
<td>1.20</td>
<td>1.67</td>
<td>15.26</td>
<td></td>
</tr>
<tr>
<td>SCAE R</td>
<td>R: VFR_real_u, real unlabeled dataset</td>
<td>9.64</td>
<td>5.73</td>
<td>10.87</td>
<td></td>
</tr>
<tr>
<td>SCAE FR</td>
<td>FR: Combination of data from F and R</td>
<td>6.52</td>
<td>2.02</td>
<td>14.01</td>
<td></td>
</tr>
</tbody>
</table>
Experiments

• Analysis of network structure
  • Table 3: Our CNN decomposition reaches its optimal performance when higher-layer convolutional filters are still trained by supervised data.
  • Table 4: A deeper $C_u$ contributes little to the results. Similar trends are observed when we fix $K$ and adjust $N$. Finally, we choose $N = 8$, $N = 2$.

| Table 3: Top-5 Testing Errors (%) for Different CNN Decompositions (Varying $K$, $N = 8$) |
|---------------------------------|---|---|---|---|---|---|
| $K$   | 0  | 1  | 2  | 3  | 4  | 5  |
| Train | 8.46 | 9.88 | 11.23 | 12.54 | 15.21 | 17.88 |
| VFR_real_test | 20.72 | 20.31 | 18.21 | 18.96 | 22.52 | 25.97 |

| Table 4: Top-5 Testing Errors (%) for Different CNN Decompositions (Varying $K$, $N = K + 6$) |
|---------------------------------|---|---|---|---|
| $K$ | 1  | 2  | 3  | 4  |
| Train | 11.46 | 11.23 | 10.84 | 10.86 |
| VFR_real_test | 21.58 | 18.21 | 18.15 | 18.24 |
Experiments

- Recognition performance on VFR datasets
  - DeepFont models fit synthetic data significantly better than LFE
  - With two font-specific augmentations, the DeepFont F adapts better to real-world data

Table 5: Comparison of Training and Testing Errors on Synthetic and Real-world Datasets (%)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Training Data</th>
<th>Training Error</th>
<th>VFR_syn_val</th>
<th>VFRWild325</th>
<th>VFR_real_test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Top-1</td>
<td>Top-5</td>
<td>Top-1</td>
</tr>
<tr>
<td>LFE</td>
<td>/</td>
<td>/</td>
<td>26.50</td>
<td>6.55</td>
<td>44.13</td>
</tr>
<tr>
<td>DeepFont S</td>
<td>/</td>
<td>F</td>
<td>0.84</td>
<td>1.03</td>
<td>64.60</td>
</tr>
<tr>
<td>DeepFont F</td>
<td>/</td>
<td>F</td>
<td>8.46</td>
<td>7.40</td>
<td>43.10</td>
</tr>
<tr>
<td>DeepFont CAE_FR</td>
<td>FR</td>
<td>F</td>
<td>11.23</td>
<td>6.58</td>
<td>38.15</td>
</tr>
<tr>
<td>DeepFont CAE_R</td>
<td>R</td>
<td>F</td>
<td>13.67</td>
<td>8.21</td>
<td>44.62</td>
</tr>
</tbody>
</table>

Figure 10: Failure VFR examples using DeepFont.
Experiments

• Evaluating font similarity
  • We use the 4096-dim output of the fc7 layer as font representation.
  • We then extract such features from all samples in VFR_syn_val dataset, obtaining 100 feature vectors per class.
  • For each class, the 100 feature vectors are averaged.
  • The Euclidean distance between the representative vectors of two font classes as their similarity measure.
Experiments

• DeepFont Model Compression
  • Since fc6 layer takes up 85% of the total model size, we focus on its compression.
  • A consistent margin of the “loss-less” method over its “lossy” counterpart.
  • It takes around 700 megabytes to store all the parameters in uncompressed DeepFont model. By further reducing the output sizes of both fc6 and fc7 to 2048, only 9,477,066 parameters are needed, causing high compression ratio of 18.73 and only 40 megabytes in storage.
Conclusion

• A large set of labeled real-world dataset as well as unlabeled real-world images are collected.
• Combat the mismatch between available training and testing data
• Model compression