Paper

- **Title**: DenseCap: Fully Convolutional Localization Networks for Dense Captioning
- **Authors**: Justin Johnson, Andrej Karpathy, Li Fei-Fei
- **Link**: http://arxiv.org/abs/1511.07571
- **Tags**: Neural Network, localization, captioning, dense captioning, detection, LSTM, R-CNN, VGG
- **Year**: 2015

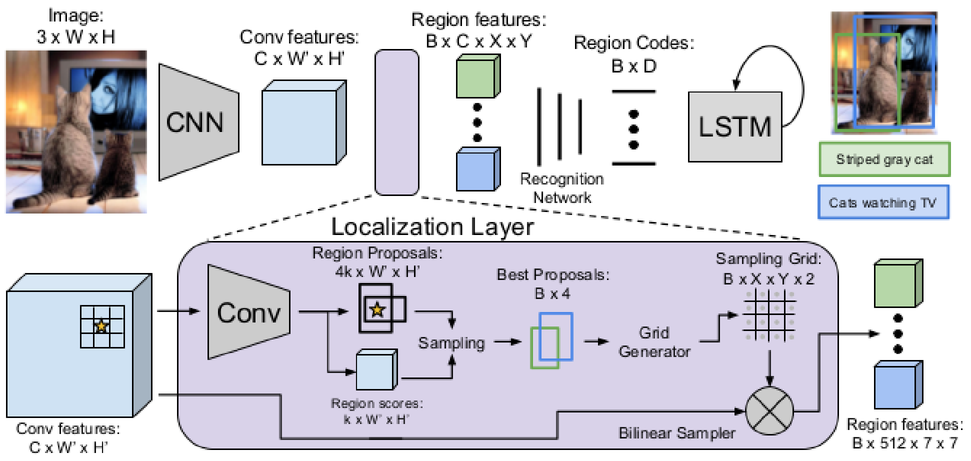
Summary

- What
  - They define four subtasks of image understanding:
    - *Classification*: Assign a single label to a whole image.
    - *Captioning*: Assign a sequence of words (description) to a whole image*
    - *Detection*: Find objects/regions in an image and assign a single label to each one.
    - *Dense Captioning*: Find objects/regions in an image and assign a sequence of words (description) to each one.
  - DenseCap accomplishes the fourth task, i.e. it is a model that finds objects/regions in images and describes them with natural language.
- How
  - Their model consists of four subcomponents, which run for each image in sequence:
    - (1) **Convolutional Network**:
      - Basically just VGG-16.
    - (2) **Localization Layer**:
      - This layer uses a convolutional network that has mostly the same architecture as in the "Faster R-CNN" paper.
      - That ConvNet is applied to a grid of anchor points on the image.
      - For each anchor point, it extracts the features generated by the VGG-Net (model 1) around that point.
      - It then generates the attributes of `k` (default: 12) boxes using a shallow convolutional net. These attributes are (roughly): Height, width, center x, center y, confidence score.
      - It then extracts the features of these boxes from the VGG-Net output (model 1) and uses bilinear sampling to project them onto a fixed size (height, width) for the next model. The result are the final region proposals.
      - By default every image pixel is an anchor point, which results in a large number of regions. Hence, subsampling is used during training and testing.
    - (3) **Recognition Network**:
      - Takes a region (flattened to 1d vector) and projects it onto a vector of length 4096.
      - It uses fully connected layers to do that (ReLU, dropout).
      - Additionally, the network takes the 4096 vector and outputs new values for the region's position and confidence (for late fine tuning).

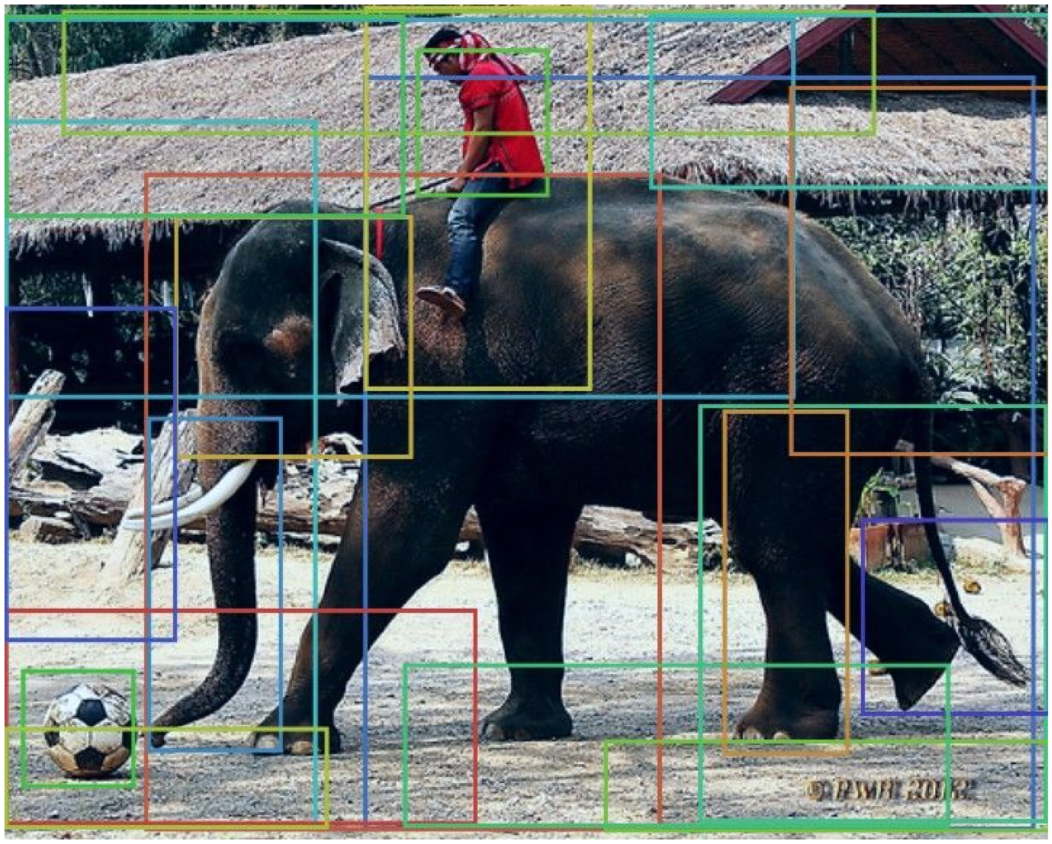
        - The 4096 vectors of all regions are combined to a matrix that is fed into the next component (RNN).
        - The intended sense of the this component seems to be to convert the "visual" features of each region to a more abstract, high-dimensional representation/description.
    - (4) **RNN Language Model**:
        - The take each 4096 vector and apply a fully connected layer + ReLU to it.
        - Then they feed it into an LSTM, followed by a START token.
        - The LSTM then generates word (as one hot vectors), which are fed back into the model for the next time step.
        - This is continued until the LSTM generates an END token.
  - Their full loss function has five components:
    - Binary logistic loss for the confidence values generated by the localization layer.
    - Binary logistic loss for the confidence values generated by the recognition layer.
    - Smooth L1 loss for the region dimensions generated by the localization layer.
    - Smooth L1 loss for the region dimensiosn generated by the recognition layer.
    - Cross-entropy at every time-step of the language model.
  - The whole model can be trained end-to-end.

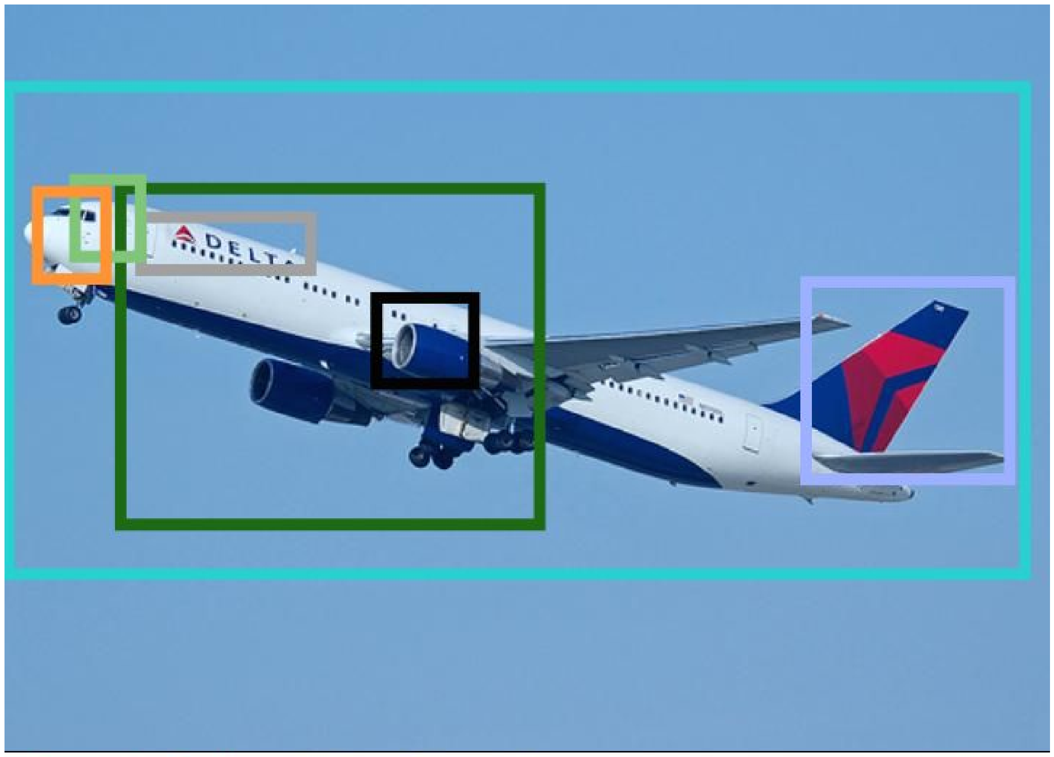
- Results

  - They mostly use the Visual Genome dataset.
  - Their model finds lots of good regions in images.
  - Their model generates good captions for each region. (Only short captions with simple language however.)
  - The model seems to love colors. Like 30-50% of all captions contain a color. (Probably caused by the dataset?)
  - They compare to EdgeBoxes (other method to find regions in images). Their model seems to perform better.
  - Their model requires about 240ms per image (test time).
  - The generated regions and captions enable one to search for specific objects in images using text queries.



Architecture of the whole model. It starts with the VGG-Net ("CNN"), followed by the localization layer, which generates region proposals. Then the recognition network converts the regions to abstract high-dimensional representations. Then the language model ("RNN") generates the caption.





plane is flying. tail of the plane. red and white plane. plane is white. engine on the plane. windows on the plane. nose of the plane.

Rough chapter-wise notes

- (1) Introduction
  - They define four subtasks of visual scene understanding:
    - Classification: Assign a single label to a whole image
    - Captioning: Assign a sequence of words (description) to a whole image
    - Detection: Find objects in an image and assign a single label to each one
    - Dense Captioning: Find objects in an image and assign a sequence of words (description) to each one
  - They developed a model for dense captioning.
  - It has two three important components:
    - A convoltional network for scene understanding
    - A localization layer for region level predictions. It predicts regions of interest and then uses bilinear sampling to extract the activations of these regions.
    - A recurrent network as the language model
  - They evaluate the model on the large-scale Visual Genome dataset (94k images, 4.1M region captions).
- (3) Model
  - Model architecture
    - Convolutional Network
      - They use VGG-16, but remove the last pooling layer.
      - For an image of size W, H the output is 512xW/16xH/16.
      - That output is the input into the localization layer.
    - Fully Convolutional Localization Layer
      - Input to this layer: Activations from the convolutional network.
      - Output of this layer: Regions of interest, as fixed-sized representations.
        - For B Regions:
          - Coordinates of the bounding boxes (matrix of shape Bx4)
          - Confidence scores (vector of length B)
          - Features (matrix of shape BxCxXxY)
      - Method: Faster R-CNN (pooling replaced by bilinear interpolation)
      - This layer is fully differentiable.
      - The localization layer predicts boxes at anchor points.
      - At each anchor point it proposes `k` boxes using a small convolutional network. It assigns a confidence score and coordinates (center x, center y, height, width) to each proposal.
      - For an image with size 720x540 and k=12 the model would have to predict 17,280 boxes, hence subsampling is used.
      - During training they use minibatches with 256/2 positive and 256/2 negative region examples. A box counts as a positive example for a specific image if it has high overlap (intersection) with an annotated box for that image.
      - During test time they use greedy non-maximum suppression (NMS) (?) to subsample the 300 most confident boxes.
      - The region proposals have varying box sizes, but the output of the localization layer (which will be fed into the RNN) is ought to have fixed sizes.
      - So they project each proposed region onto a fixed sized region. They use bilinear sampling for that projection, which is differentiable.
    - Recognition network
      - Each region is flattened to a one-dimensional vector.
      - That vector is fed through 2 fully connected layers (unknown size, ReLU, dropout), ending with a 4096 neuron layer.
      - The confidence score and box coordinates are also adjusted by the network during that process (fine tuning).
    - RNN Language Model

        - Each region is translated to a sentence.
        - The region is fed into an LSTM (after a linear layer + ReLU), followed by a special START token.
        - The LSTM outputs multiple words as one-hot-vectors, where each vector has the length `V+1` (i.e. vocabulary size + END token).
        - Loss function is average crossentropy between output words and target words.
        - During test time, words are sampled until an END tag is generated.
  - Loss function
    - Their full loss function has five components:
      - Binary logistic loss for the confidence values generated by the localization layer.
      - Binary logistic loss for the confidence values generated by the recognition layer.
      - Smooth L1 loss for the region dimensions generated by the localization layer.
      - Smooth L1 loss for the region dimensiosn generated by the recognition layer.
      - Cross-entropy at every time-step of the language model.
    - The language model term has a weight of 1.0, all other components have a weight of 0.1.
  - Training an optimization
    - Initialization: CNN pretrained on ImageNet, all other weights from `N(0, 0.01)`.
    - SGD for the CNN (lr=?, momentum=0.9)
    - Adam everywhere else (lr=1e-6, beta1=0.9, beta2=0.99)
    - CNN is trained after epoch 1. CNN's first four layers are not trained.
    - Batch size is 1.
    - Image size is 720 on the longest side.
    - They use Torch.
    - 3 days of training time.

- (4) Experiments

  - They use the Visual Genome Dataset (94k images, 4.1M regions with captions)
  - Their total vocabulary size is 10,497 words. (Rare words in captions were replaced with `<UNK>`.)
  - They throw away annotations with too many words as well as images with too few/too many regions.
  - They merge heavily overlapping regions to single regions with multiple captions.
  - Dense Captioning
    - Dense captioning task: The model receives one image and produces a set of regions, each having a caption and a confidence score.
    - Evaluation metrics
      - Evaluation of the output is non-trivial.
      - They compare predicted regions with regions from the annotation that have high overlap (above a threshold).
      - They then compare the predicted caption with the captions having similar METEOR score (above a threshold).
      - Instead of setting one threshold for each comparison they use multiple thresholds. Then they calculate the Mean Average Precision using the various pairs of thresholds.
    - Baseline models
      - Sources of region proposals during test time:
        - GT: Ground truth boxes (i.e. found by humans).
        - EB: EdgeBox (completely separate and pretrained system).
        - RPN: Their localization and recognition networks trained separately on VG regions dataset (i.e. trained without the RNN language model).
      - Models:
        - Region RNN model: Apparently the recognition layer and the RNN language model, trained on predefined regions. (Where do these regions come from? VG training dataset?)

      - Full Image RNN model: Apparently the recognition layer and the RNN language model, trained on full images from MSCOCO instead of small regions.
      - FCLN on EB: Apparently the recognition layer and the RNN language model, trained on regions generated by EdgeBox (EB) (on VG dataset?).
      - FCLN: Apparently their full model (trained on VG dataset?).
  - Discrepancy between region and image level statistics
    - When evaluating the models only on METEOR (language "quality"), the *Region RNN model* consistently outperforms the *Full Image RNN model*.
    - That's probably because the *Full Image RNN model* was trained on captions of whole images, while the *Region RNN model* was trained on captions of small regions, which tend to be a bit different from full image captions.
  - RPN outperforms external region proposals
    - Generating region proposals via RPN basically always beats EB.
  - Our model outperforms individual region description
    - Their full jointly trained model (FCLN) achieves the best results.
    - The full jointly trained model performs significantly better than `RPN + Region RNN model` (i.e. separately trained region proposal and region captioning networks).
  - Qualitative results
    - Finds plenty of good regions and generates reasonable captions for them.
    - Sometimes finds the same region twice.
  - Runtime evaluation
    - 240ms on 720x600 image with 300 region proposals.
    - 166ms on 720x600 image with 100 region proposals.
    - Recognition of region proposals takes up most time.
    - Generating region proposals takes up the 2nd most time.
    - Generating captions for regions (RNN) takes almost no time.
- Image Retrieval using Regions and Captions
  - They try to search for regions based on search queries.
  - They search by letting their FCLN network or EB generate 100 region proposals per network. Then they calculate per region the probability of generating the search query as the caption. They use that probability to rank the results.
  - They pick images from the VG dataset, then pick captions within those images as search query. Then they evaluate the ranking of those images for the respective search query.
  - The results show that the model can learn to rank objects, object parts, people and actions as expected/desired.
  - The method described can also be used to detect an arbitrary number of distinct classes in images (as opposed to the usual 10 to 1000 classes), because the classes are contained in the generated captions.