Differential Attention for Visual Question Answering

Paper: https://arxiv.org/abs/1804.00298
Code: https://github.com/chirag26495/DAN_VQA

Team - 13:

- Chirag Parikh - 2022900005
- Neeraj Veerla - 2021121008
- Adhiraj Deshmukh - 2021121012
- Shreya Patil - 2021121009

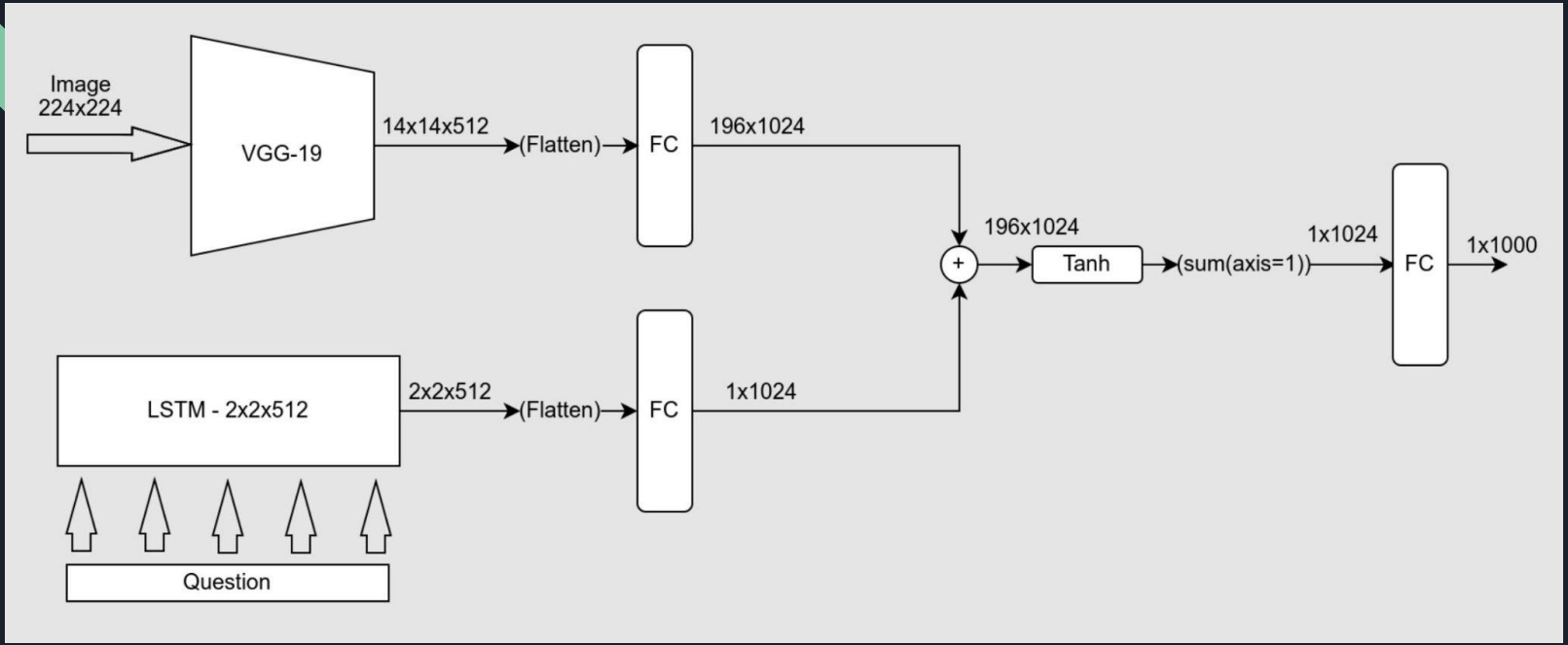
Key Components

- Basic VQA model [LSTM Q + CNN I]
- Finding Exemplars [KD-Tree and k-means clustering]
- Baseline: Simple Attention model [LSTM Q + CNN I + Attention]
- Paper: Differential Attention model [DAN + LSTM Q + CNN I + Attention]
- Other: Stacked Attention Network [SAN-2]

VQA-v2 Dataset

- Contains open-ended questions about images which require an understanding of vision, language and common sense knowledge to answer.
- 265,016 images (COCO and abstract scenes)
- At least 3 questions (5.4 questions on average) per image
- 10 ground truth answers per question
- Total dataset size: 14GB
- Training Split: 4,43,757 (I+Q pairs)
- Validation Split: 2,14,354 (I+Q pairs)

Basic VQA model (LSTM Q + CNN I)

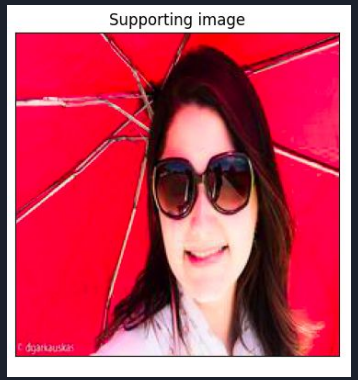


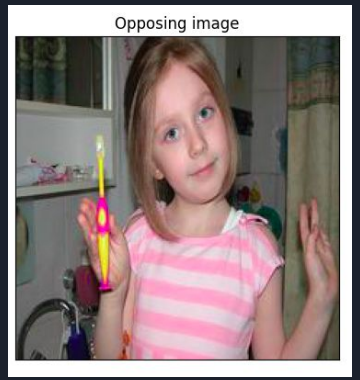
- **Joint image-question level embedding** from the above model is used to relate meaningful exemplars.
- Output: Train features of (4,43,757 x 1024) and Val features of (2,14,354 x 1024)

Finding Supporting and Opposing Exemplars

Author's method

- For each data point in the embedding space, find the closest neighbours using KD-tree.
- Once the neighbours are found, perform clustering of the neighbours.
- Choose the nearest clusters as supporting and farthest clusters as opposing.
- The main idea was not choosing opposing examples too far from the data point.





Finding Supporting and Opposing Exemplars (cont.)

- The method proposed by the author is computationally expensive.
  Estimated runtime of 59 days on ADA, with time complexity of
  $$O(D . [N . \log(|D|) + N . k . t])$$
- Alternative method:
  - We choose the nearest 50 data-points as supporting exemplars,
  - 1200-1500 nearest neighbours for opposing exemplars.
  - Reduces time complexity to $O(N . D . \log|D|)$
    Reduces the time of computation by a factor of $\approx 10^2$
  - With multiprocessing, reduced the time to ~12hrs

Time for computation per instance:

Method	Query-time	k-means	compute-distance	Total
KNN + K-Means	0.7231 s	11.8403 s	0.0002 s	12.75 s
KNN	0.583 s	-	0.0002 s	0.591 s

Finding Supporting and Opposing Exemplars (cont.)

Alternative method:

- We first clustered the whole data using Mini Batch K-means using multiprocessing.
- Performed KD-Tree on the cluster centres of the data-points.
- For all the points in a particular cluster, we choose the nearest clusters to be the supporting and the clusters in range 20 to 25 to be opposing clusters.
- This method was tried out to reduce the time complexity of the find examples in just 10 mins.

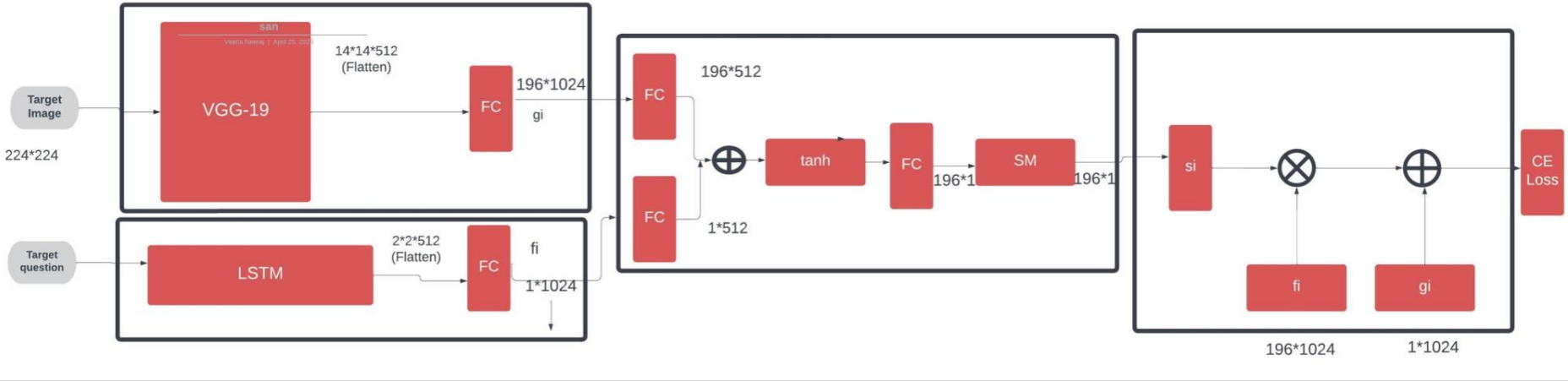
Where are these people sitting?

grass, on grass, park

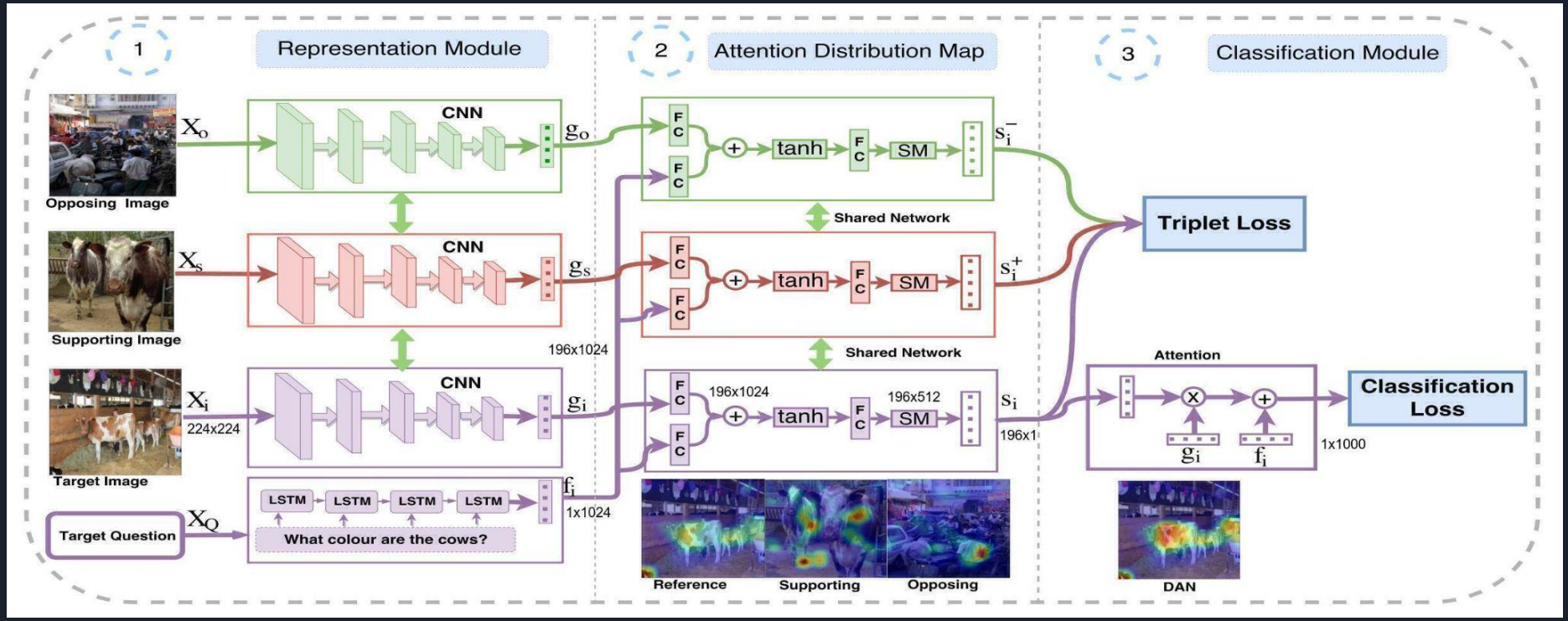
Supporting image

Opposing image

Baseline: (LSTM Q + CNN I + Attention)



Differential Attention Network (DAN) + Baseline



Triplet Loss

- Alpha margin (=0.2) was empirically chosen, on the basis of loss on validation dataset.
- Si, Si+, Si- are the attention weighted regions of target image, supporting exemplar and opposing exemplar.

$$T(s_i, s_i^+, s_i^-) = \max(0, ||t(s_i) - t(s_i^+)||_2^2 + \alpha - ||t(s_i) - t(s_i^-)||_2^2)$$

Cross-Entropy Loss

- C (=1000) is no. of answer categories.
- v (=1) is the weightage of Triplet Loss in the Total loss.

$$L(\mathbf{s}, \mathbf{y}, \theta) = \frac{1}{N} \sum_{i=1}^{N} (L_{cross}(\mathbf{s}, \mathbf{y}) + \nu T(s_i, s_i^+, s_i^-))$$

$$L_{cross}(\mathbf{s}, \mathbf{y}) = -\frac{1}{C} \sum_{j=1}^{C} y_j \log p(c_j|\mathbf{s})$$

Experiments performed on DAN

Attention Score in loss function:

We tried different notions of attention score that contribute to the triplet loss.

- **ha = torch.tanh(hi+hq)**
  - Performs the best among all, this is the concat of image and question and tanh
- **pi = torch.softmax(ha, dim=1)**
  - Gives close performance, but the triplet loss doesn't converge or change.
- **vi_attended = (pi * vi).sum(dim=1)**
  - This, the pi attention scores weighted accordingly to the visual features.
- **u = vi_attended + vq**
  - Adding question feature to the attended image features.

Hyperparameter tuning

Margin:

- The margin helps in defining a decision boundary for the embeddings in the feature space.
- The model learns to keep the positive examples within a certain distance from the anchor example and the negative examples beyond a certain distance from the anchor example.
- This ensures that the embeddings of examples of the same class are closer together and those of different classes are farther apart in the feature space.

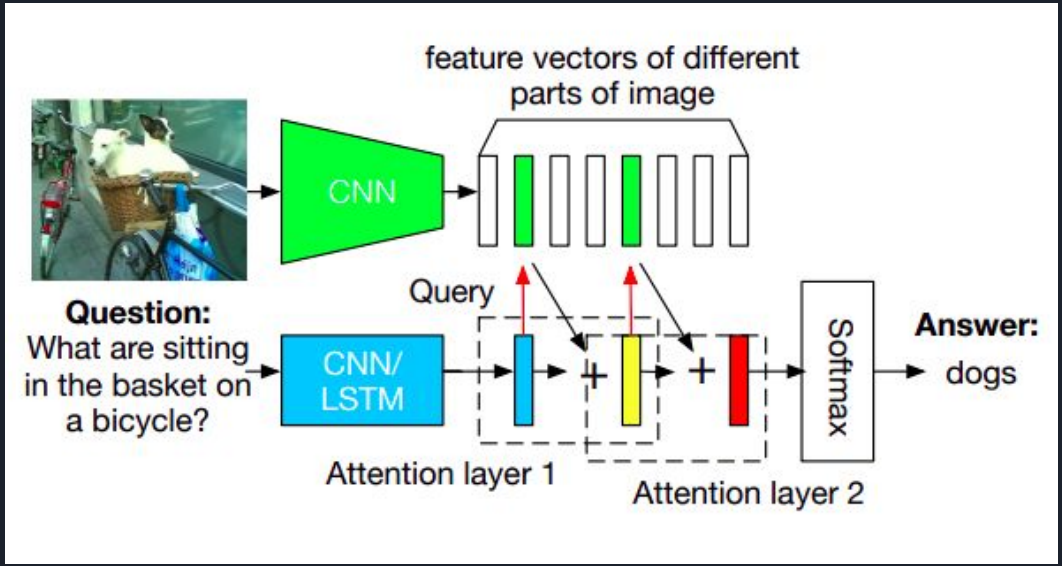
We tried to increase the margin from 0.2 to 0.8, **0.2** works the best, this is also dependent on how we find the exemplars and the parameters used in the exemplar finding.

V Weightage:

- Weighs the importance of triplet loss as compared to the CE loss, in the final loss of the model.

From 1 to 10, **10** works fine, as triplet loss decrease rapidly and the value is small.

Other model: Stacked Attention Networks (SAN-2)



- Stacking an additional Attention layer over our Baseline model (using updated question features) for SAN-2.

Quantitative Results

Models	All	Yes/No	Number	Other
Basic	47.61	74.86	37.21	29.61
SAN-1	53.23	76.39	38.53	39.49
SAN-2	55.28	77.15	**39.69**	**42.76**
DAN	**55.49**	**77.23**	39.59	42.55
DAN-alt.	54.16	77.13	38.75	40.77

Qualitative Results