Learning Video Features for Multi-Label Classification

Shivam Garg[0000−0002−7213−3967]

shivgarg@live.com

Abstract. This paper studies some approaches to learn representation of videos. This work was done as a part of Youtube-8M Video Understanding Challenge. The main focus is to analyze various approaches used to model temporal data and evaluate the performance of such approaches on this problem. Also, a model is proposed which reduces the size of feature vector by 70% but does not compromise on accuracy.
The first approach is to use recurrent neural network architectures to learn a single video level feature from frame level features and then use this aggregated feature to do multi-label classification. The second approach is to use video level features and deep neural networks to assign the labels.

Keywords: RNN, LSTM, MoE, ResidualCNN

1 Introduction

Video classification is one of the most important problem in computer vision. Video content consists of temporally related images. This temporal dimension adds a new level of complexity to the image classification problem. In recent years, with the advent of faster computational platforms alongwith high quality large datasets like Imagenet[4], MS COCO[14], Pascal VOC[5], robust image classfication and detection algorithms have been developed. Human level performance on Imagenet was surpassed in 2015 by ResNet[8]. Many object detection approaches use some of the highly performant approaches on imagenet as a backbone network for eg. Faster RCNN[19], Yolo[18], SSD[15].This approach of transfer learning aids the object detection a lot. Similarly development of good video classification models would lead to improvement in a lot of tasks like tracking object movement in videos, detecting suspicious activity, video captioning, summarisation , robotic vision, affective computing, HCI etc.
Understanding a video involves two parts, understanding what is happening in a single frame and correlating the information present in various frames. The harder part lies in the correlation of information in different frames. LSTM's[24] have shown promising results in natural language modelling problems which exhibit similar difficulties. Thus a similar approach is used here.
For the former part of extracting features from a single frame, approaches based on deep convolution networks [8,20,21] are good at capturing salient features

outperforming many hand crafted techniques [3,16,17,2] on various computer vision tasks.
The paper is divided into the following sections : Section 2 describes the dataset, Section 3 describes the evaluation criteria, Section 4 describes the models, Section 5 presents the experimental results and Section 6 concludes the work.

2 Dataset

The dataset used for analysis is the Youtube-8M[1]. This is by far the largest dataset for video classification available to date. Other video datasets include Sports-1M[12] and ActivityNet[6] .These datasets are limited to sports videos and human activity recognition tasks respectively.
Youtube-8M contains 6.1 million videos, which have been roughly partitioned in 3.84 million examples for training and 1.13 million each for validation and test set. The total number of labels is 3862 across 24 top level entities in knowledge graph. The average number of labels per video is 3.
The dataset is divided into two parts, frame-level and video-level.The frame-level features contain features of upto 360 frames per video. The frames are processed at 1 fps and a maximum of first 6 minutes of each video is processed. Each frame has a 1024 length vector containing rgb features and 128 length vector containing audio features. The visual features have been extracted from Inception V3[22]. The features are extracted just after the last global pooling layer .The audio features have been extracted using a model similar to VGG as described in Hershey et. al. [9]. The visual features extracted from the inception network are further processed. The extracted feature length is 2048 which is reduced to 1024 by PCA. Both audio and visual features are quantized to 8bits to reduce the size of the features. The data is presented in 3844 shards of equal sizes for each of the train,test and validation split. The total size of frame-level features is 1.53TB.
The video-level dataset is similar. The rgb and audio features are mean aggregated frame level features. The feature values are not quantized in contrast to frame level features. The dataset is formatted in tfrecord format. Similar to frame level, it is provided in 3844 equally sized shards for each train,validation and test split.The total size of video-level features is 31GB.
The data has a lot of challenging aspects which are listed here under :-

1. Scale :- As mentioned above, there are approx. 5 million videos in the train+validation set of the dataset. Processing all the videos for some frame-level models takes a lot of time(in order of days). So GPUs with large amount of memory, alongwith high speed storage device is essential for quick experimentation and hyperparameter tuning.
2. Noisy labels: The dataset is annotated by youtube annotation system which results in missing labels. This makes training even harder with model getting contradictory signals for similar inputs.
3. Imbalance: The data is highly imbalanced with a small number of label entities having a majority of videos. The log log plot of number of videos for each entity follows a zipf(Fig. 1) distribution, depicting the imbalance in the

data distribution. This imbalance in the dataset makes it difficult to learn about rare classes, leading to poor performance over those classes. This fact is supported by low MAP values obtained by the models.

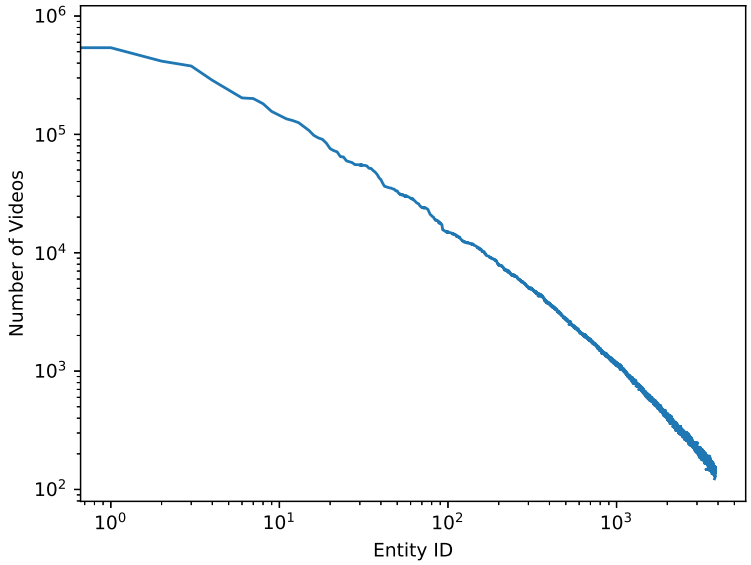

Fig. 1. log log plot of number of videos for a single entity

As the rgb and audio information is provided in encoded form, the focus of the problem shifts to analysis of temporal modeling approaches for given features.

3 Evaluation

The following metrics have been used to evaluate the performance of a model.

1. GAP@20 :- Global Average Precision is the primary evaluation metric used in the paper. For each video, a list of labels and the confidence scores is calculated. Then top 20 labels and the confidence scores are picked and added as individual data points to a global pool. Then this pool of label-confidence pairs is sorted and the average precision is calculated over this pool.

$$GAP = \sum_{i=0}^{T} [p(i)\{r(i+1) - r(i)\}] \tag{1}$$

where p(i) denotes the precision of the first i examples, and r(i) denotes the recall till the first i examples.
2. MAP :- Mean Average Precision is the mean of average precision of each class. For each class, average precision of the predictions for that class is calculated. The mean of these values for all classes is MAP.
3. PERR :- Precision at Equal Recall Rate. To calculate this metric, top k labels for each video are extracted where k is number of labels in the ground truth. Then precision is calculated for the extracted labels and the calculated precision values are averaged for all the videos in the dataset.
4. Hit@1 :- For each video, the most confident label is picked and checked whether it belongs to the ground set labels. If it belongs, a score of 1 is assigned otherwise a 0. Then scores for all videos are averaged to calculate the final score.

4 Models

In this section, models are described. There are two categories, one category exploring frame-level features and other exploring video-level features.

4.1 Frame Level Models

The models here exploit the structure in per frame features. The common structure of all the models in this section is to aggregate frame level features into a single feature vector per video and then pass this feature vector to a mixture of experts model for determining the labels. The basic network architecture used in the models is RNN.

LSTM Recurrent Neural Networks are designed to handle temporal data.In this work, LSTM[24] cell is used in all RNN's. Long Short Term Memory is one of the most successful tool in solving various NLP problems. They have been used in a variety of problems like machine translation, image captioning, text to speech, speech to text, etc. LSTMs are designed to learn long term dependencies. LSTM cell maintains a cell state, which helps the cell to remember relevant information from all the preceding steps. A basic LSTM cell operation can be summarized in four steps:

1. Calculate the forget_gate which determines the part of the cell state that should be remembered and the part that should be forgotten.

$$f_t = \sigma\left[W_f.(h_{t-1} \| x_t) + b_f\right] \tag{2}$$

2. Filter out relevant information from input at step t, which is used to update the cell state.(input_gate).

$$i_t = \sigma\left[W_i.(h_{t-1} \| x_t) + b_i\right] \tag{3}$$

$$\hat{c}_t = tanh\left[W_c.(h_{t-1} \| x_t) + b_c\right] \tag{4}$$

3. Update the cell state by using the forget_gate and input_gate

$$c_t = \sigma\left[f_t * c_{t-1} + i_t * \hat{c}_t\right] \tag{5}$$

4. Determine the parts of the cell state which should be given as output of the cell(output_gate).

$$o_t = \sigma\left[W_o.(h_{t-1} \| x_t) + b_o\right] \tag{6}$$
$$h_t = o_t * tanh(c_t) \tag{7}$$

In the above equations,|| denotes concatenation operation, * denotes elementwise multiplication.
The first set of models use N layer LSTM architecture which is followed by a pooling layer to pool the outputs of each lstm cell. The pooled feature is then passed into a MoE model to determine the labels(Fig. 2). Three different pooling

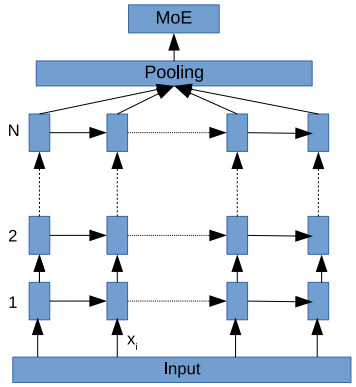

Fig. 2. N layer LSTM model with a pooling module and a final mixture of experts layer to produce outputs.

mechanisms are used :-

1. Choosing the last state of lstm .
2. Maxpooling all the feature vectors
3. Applying attention weighting to input feature vectors.

All frames in a video are not equally important for determining the semantic content in the video. For eg., the starting and ending frames do not contain much information, dark frames in a video do not convey much or in a talk show most of the frames are similar, it is the audio content that describes the genre of video. Attention module can help to solve this problem which weighs the feature vectors at all time steps according to the importance of the feature vector for classification task. Then a weighted sum of these feature vectors is used as

an input for the MoE model. The method of calculating weighted vector(F) is described here under:-

$$I = tanh\,(H) \tag{8}$$
$$w = softmax\,[I.W_a] \tag{9}$$
$$F = tanh(reduce\_sum(w.H)) \tag{10}$$

H is the output of lstm network at all time steps. W_a is a learnable column vector, which weighs the vectors.
Deeper LSTM models are harder to optimise due to introduction of a lot of non-linearities, and they might suffer from vanishing or exploding gradient problems. Residual connections introduced in ResNet[8] come handy to solve some of the problems in optimizing such networks. The residual connections force a part of the network to learn h(x) -x , where h is the hypothesis function to be learnt and x is the interim representation learnt.This modeling trick facilitates the flow of gradients in deep networks. Residual connections are utilized in lstm network with a depth of 3. The network architecture is shown in Fig. 3.

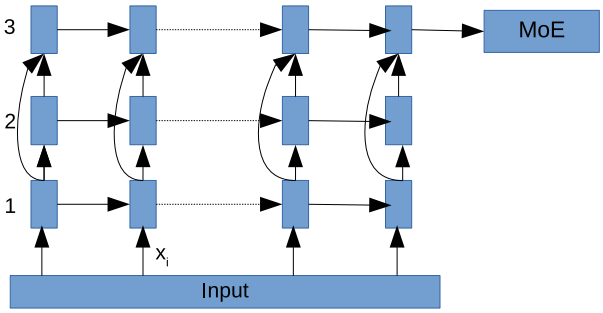

Fig. 3. 3 layer LSTM model with residual connections.

BiLSTM BiLSTM is similar to LSTM, with the difference being BiLSTM contains two LSTM chains, one of which processes the data in forward direction and the other processes the data in backward direction. BiLSTMs try to learn representations by considering temporal relations in both directions. A lot of times, context from future frames is helpful in understanding the present frame, which BiLSTMs model efficiently.The basic architecture is depicted in Fig. 4. The outputs of both the forward and backward lstms is concatenated to produce the output for BiLSTM. Other approaches like mean,max aggregation, attention weighting can be applied to produce the output vector.
A variant of BiLSTM is also proposed here. Currently, the forward and backward lstms do not interact with each other throughout the depth of BiLSTM.This limits the ability of network to capture various dependencies for eg. when a state

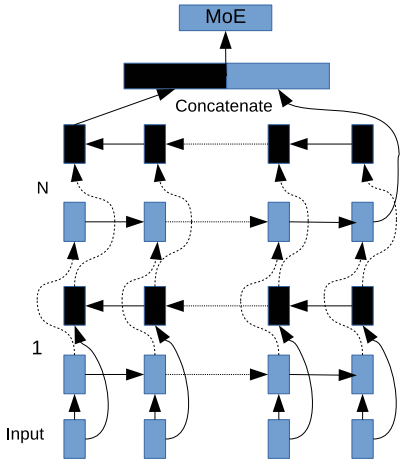

Fig. 4. N layer BiLSTM model

depends on both past and future states. In this N layer network, the forward and backwards outputs at each time step of every layer are pooled before passing them as input to upper layers. It can be thought of as N single BiLSTM layers stacked over one another, where the input to the n^{th} layer will be the pooled output of $n-1^{th}$ layer at each time step. Since the network contains N single BiLSTM stacked over one another, it would be referred to as Stacked BiLSTM in this paper.

Multi-modal Approaches The data contains rgb and audio features for each frame. Therefore, various multi-modal techniques can be adopted to model rgb and audio features separately. The audio and rgb features may have different distributions and independent models for each data stream can help to capture such relations efficiently. A simple framework as shown in Fig. 5 was adopted to judge the effectiveness of such techniques. The details of the model used is given in the experiments section.

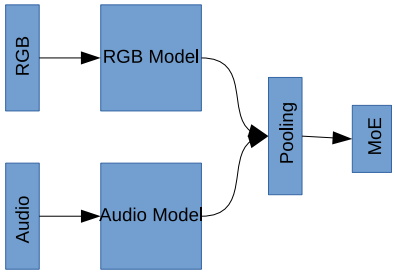

Fig. 5. Multi Modal architecture

4.2 Video-Level Models

The models is this section use the mean aggregated frame level features. The models were approached with the motive of learning a smaller representation of video-level features without sacrificing on accuracy.

ResidualCNN-X The inspiration to try deep convolutional neural networks comes from the success of deep CNN on image classfication tasks especially on Imagenet Competition. Even though the features here lack explicit spatial relations, CNN's perform well for various natural language problems where the inputs lack explicit spatial relations and exhibit temporal relations as shown in Kim et. al[13].
The network is made of two parts. The first part is a fully connected layer and the second part is a deep cnn network. The deep cnn network is made up of several residual modules(Fig. 6). The residual module is made up of dilated convolution layers with kernel size (9,1) and stride=1.The dilation of conv layer depends on the position in residual module where the conv layer is located. The ith layer of residual module has a dilation of $2 * i + 1$ with index starting from 0. Dilation helps the cnn to cover large parts of input length. Each convolution layer has same padding, which keeps the output dimension equal to input dimension. The number of channels in convolution layers of residual modules varies with each instance of module.

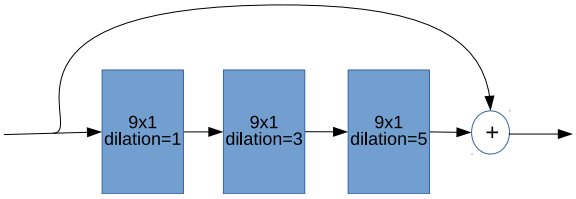

Fig. 6. A 3 layer residual module

The depth of residual modules used in ResidualCNN-X is atmost 3. All the convolution layers use ReLu activation. Batch normalisation[11] has been used which helps in faster and stable training. Batch normalization layer is added after layers specified in Table 1. Mean and max aggregation pooling methods have been used to pool the output of the deep cnn network. X in ResidualCNN-X denotes a hyperparameter which specifies the length of output vector used by MoE for label assignment. Therefore, ResidualCNN-1024 would mean that the output of network is a 1024 dimensional vector which is then passed to MoE for label classification. The network lacking the deep neural network, containing only the fully connected layer would be known as ResidualCNN-FC-X. X would carry similar meaning for this network too. Refer to Fig. 7 and Table 1 for full details of ResidualCNN-X architecture.

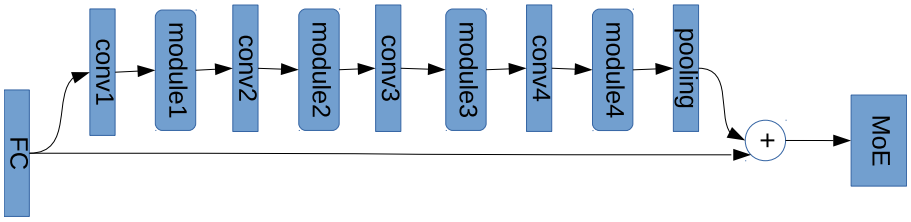

Fig. 7. ResidualCNN-X architecture

Table 1. ResidualCNN-X architecture details

Layer	kernel size	depth [1]	output size	9x1,dil=1	9x1,dil=3	9x1,dil=5	batch norm
FC			Xx1x1				
conv1	49x1	64	Xx1x64				y
module1		3	Xx1x64	128	192	64	y
conv2	1x1	128	Xx1x128				y
module2		3	Xx1x128	256	512	128	y
conv3	1x1	256	Xx1x256				y
module3		2	Xx1x256	512	256		y
conv4	1x1	X	XxlxX				y
module4		2	XxlxX	512	X		y
pooling			Xx1x1				

[1] num. of channels for conv layer, num. of conv layer for residual module.

5 Experiments & Results

This section contains details of experiments and the results obtained. The details common to all experiments are:-

- The input to the models was the concatenation of rgb and audio features resulting in 1152 length feature vector except multi-modal approaches.
- Adam optimizer was used to optimize the models, with learning rate decay rate of 0.95 every epoch.
- Cross entropy loss was used as the loss function for model optimization.
- All the models were trained on the train split of the dataset and validation set was used to evaluate the performance of models.

The accuracy measures on validation set followed closely(within 0.2%) the results on test dataset as reported by the kaggle leaderboard. Since test labels were not available, to compute all performance measures, validation set is used. The compute platform was Nvidia GTX 1080Ti graphic card, Ubuntu 16.04 OS, 8GB RAM and a 2TB external storage. Due to limitation of computational resources, the experiments were limited to maximize single model

accuracy. Therefore ensemble/bagging/boosting/stacking methods were not explored which would have boosted the performance of existing models. The results are provided in three sections,first for frame-level models, second for video-level models and the last analysing feature compression. The code base is uploaded at `https://github.com/shivgarg/youtube-8m`. The trained model files are also uploaded to Onedrive with details given in github repository.

5.1 Frame-level-models

The models have been described in section 4.1. All the models were trained for 2 epochs. Learning rate of 0.001 was used for all models. The training time ranged from 1 day for smaller models to about 3 days for large models. The details of models, their hyperparameters and accuracy values is given in Table:2. (LSTM

Table 2. Frame-level model hyperparameters and results

Model	Batch Size	Hit@1	PERR	MAP	GAP
LSTM 1 layer, 1024 [1]	256	0.848	0.741	0.334	0.805
LSTM 2 layer, 1024	256	0.859	0.754	0.354	0.816
LSTM 2 layer, 1280,640 [2]	256	0.862	0.759	0.357	0.820
LSTM 3 layer, 1024	128	0.874	0.776	0.399	0.837
LSTM 3 layer, res[3], 1024	128	**0.877**	**0.780**	0.422	**0.842**
LSTM 1 layer, 1024, max-pooling	256	0.844	0.735	0.341	0.795
LSTM 1 layer, 1024, att-pooling[4]	256	0.863	0.761	0.378	0.822
BiLSTM 1 layer, 2048	256	0.874	0.778	**0.437**	0.839
BiLSTM 2 layer, 2048	128	0.869	0.771	0.409	0.833
Stacked BiLSTM 2 layer, 2048	128	**0.876**	**0.780**	0.427	**0.841**
BiLSTM 3 layer, 1024	128	0.866	0.763	0.359	0.824
Stacked BiLSTM 3 layer, res[3], 1024	128	0.866	0.752	0.358	0.828
Multi-Modal, DBoF, DBoF	512	0.854	0.747	0.353	0.800
Multi-Modal, (LSTM 2,1024), (LSTM 2,1024)	128	0.856	0.752	0.358	0.814

[1] LSTM output feature size
[2] LSTM 2 layer with hidden state sizes of 1280 and 640
[3] Residual connections as in Fig. 3
[4] Attention mechanism as defined in Eq. 8,9,10

3 layer, res, 1024) model and (Stacked BiLSTM 2 layer, 2048) perform the best, beating all other models in three metrics. It is closely followed by (LSTM 3 layer, 1024) and (BiLSTM 1 layer,2048). Some general trends observed :-

- Deeper networks lead to better accuracy.
- Residual connections help to optimize deeper networks.
- Stacked BiLSTM perform better than simple BiLSTM.

– Attention pooling boosts the performance of LSTM.

The performance of each of the above models can be increased by training for larger number of iterations.

5.2 Video-level models

The models have been described in 4.2. Learning rate for MoE model was 0.01, and ResidualCNN-X was 0.0001. Batch size was 1024 for MoE, and 256 for ResidualCNN-X models. The main aim of these models was to learn condensed representation without compromising with accuracy. The details of experiments, model hyperparameters are given in Table:3.
ResidualCNN-1024 performs the best in video-level models, beating the baseline

Table 3. Video-level model hyperparameters and results

Model	Batch Size	Hit@1	PERR	MAP	GAP
MoE	1024	0.854	0.759	**0.466**	0.817
ResidualCNN-FC-320	1024	0.832	0.726	0.348	0.782
ResidualCNN-FC-1024	1024	0.840	0.735	0.374	0.795
ResidualCNN-320, Mean-pooling	256	0.851	0.724	0.304	0.796
ResidualCNN-320, Max-pooling	256	0.862	0.764	0.418	0.822
ResidualCNN-512, Mean-pooling	256	0.857	0.758	0.408	0.818
ResidualCNN-512, Max-pooling	256	0.864	0.766	0.428	0.827
ResidualCNN-1024, Mean-pooling	256	0.861	0.764	0.425	0.825
ResidualCNN-1024, Max-pooling	256	**0.863**	**0.768**	0.427	**0.831**

by about 1.4%. ResidualCNN-X models tend to perform better than baseline demonstrating the effectiveness of deeper neural networks. Also, max-pooled variants performed consistently better than their mean pooled counterparts.

5.3 Feature Compression

This section highlights some models, which learnt video representations significantly smaller than the one presented and still achieved better accuracy numbers than the baseline(MoE). ResidualCNN-320, MaxPool reduces the feature size by 72% without losing on the performance metrics. This network can be used to further encode the features, which can be used by other models to predict labels.
Another direction for reducing the feature size is to explore the autoencoder category of models which perform well on the task of reducing the size of input representation and capturing salient features in the input.

Table 4. Feature Compression Results

Model	Feature Size	GAP	Δ GAP (in%)[1]	Δ Feature size [2]
LSTM 2 layer, (1280,640)	640	0.820	0.3%	0.44
ResidualCNN-512, Max Pooling	512	0.827	1%	0.55
ResidualCNN-320, Max Polling	320	0.822	0.5%	0.72

[1] (GAP(Model) - GAP(MoE))
[2] (1152 - Feature_size)/1152

6 Conclusion

In this paper, a lot of techniques were used to learn features of video for classification task. LSTM and it variations were used to model the video features for the classification task. A deep convolution neural network architecture is also proposed which helps in reducing the feature size. Deeper neural networks , both LSTM's and CNN's can give improved resutls. Transformer[23] based approaches may perform good as they show promising results on machine translation task. Ensemble methods can improve the accuracy of the approaches mentioned in this paper. Techniques such as model distillation[10] can be used to transfer the knowledge from large models to smaller models to increase the accuracy of smaller models which are more scalable and efficient. Various network structure optmisiation strategies like weights quantization, pruning networks[7] etc. can be used to reduce the size and complexity of large models.

References

1. Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S.: Youtube-8m: A large-scale video classification benchmark. CoRR **abs/1609.08675** (2016), `http://arxiv.org/abs/1609.08675`
2. Chandrasekhar, V., Takacs, G., Chen, D., Tsai, S., Grzeszczuk, R., Girod, B.: Chog: Compressed histogram of gradients a low bit-rate feature descriptor. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 2504–2511. IEEE (2009)
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. vol. 1, pp. 886–893. IEEE (2005)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
5. Everingham, M., Eslami, S.M.A., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision **111**(1), 98–136 (Jan 2015)
6. Fabian Caba Heilbron, Victor Escorcia, B.G., Niebles, J.C.: Activitynet: A large-scale video benchmark for human activity understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 961–970 (2015)

7. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
8. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. CoRR **abs/1502.01852** (2015), http://arxiv.org/abs/1502.01852
9. Hershey, S., Chaudhuri, S., Ellis, D.P.W., Gemmeke, J.F., Jansen, A., Moore, C., Plakal, M., Platt, D., Saurous, R.A., Seybold, B., Slaney, M., Weiss, R., Wilson, K.: Cnn architectures for large-scale audio classification. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2017), https://arxiv.org/abs/1609.09430
10. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
11. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
12. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR (2014)
13. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
14. Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. CoRR **abs/1405.0312** (2014)
15. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
16. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
17. Mikolajczyk, K., Schmid, C.: Indexing based on scale invariant interest points. In: Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on. vol. 1, pp. 525–531. IEEE (2001)
18. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
19. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR **abs/1506.01497** (2015), http://arxiv.org/abs/1506.01497
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
21. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the impact of residual connections on learning. corr abs/1602.07261. URL http://arxiv.org/abs/1602.07261 (2016)
22. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2818–2826 (2016)
23. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. CoRR **abs/1706.03762** (2017), http://arxiv.org/abs/1706.03762
24. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)