Learning Effective Visual Relationship Detector on 1 GPU

Yichao Lu *
Layer6 AI
yichao@layer6.ai

Cheng Chang *
Layer6 AI
jason@layer6.ai

Himanshu Rai *
Layer6 AI
himanshu@layer6.ai

Guangwei Yu
Layer6 AI
guang@layer6.ai

Maksims Volkovs
Layer6 AI
maks@layer6.ai

Abstract

We present our winning solution to the Open Images 2019 Visual Relationship challenge. This is the largest challenge of its kind to date with nearly 9 million training images. Challenge task consists of detecting objects and identifying relationships between them in complex scenes. Our solution has three stages, first object detection model is fine-tuned for the challenge classes using a novel weight transfer approach. Then, spatio-semantic and visual relationship models are trained on candidate object pairs. Finally, features and model predictions are combined to generate the final relationship prediction. Throughout the challenge we focused on minimizing the hardware requirements of our architecture. Specifically, our weight transfer approach enables much faster optimization, allowing the entire architecture to be trained on a single GPU in under two days. In addition to efficient optimization, our approach also achieves superior accuracy winning first place out of over 200 teams, and outperforming the second place team by over 5% on the held-out private leaderboard.

1. Introduction

Visual relationship detection is a core computer vision task that has gained a lot of attention recently [4, 12, 13, 14]. The task comprises of object detection followed by visual relationship prediction to identify relationships between pairs of objects. Relationship identification involves inferring complex spatial, semantic and visual information between objects in a given scene, which is a challenging task. Successfully solving this task is a natural first step towards scene understanding and reasoning. The Open Images 2019 Visual Relationship challenge introduces a uniquely large and diverse dataset of annotated images designed to benchmark visual relationship models in a standardised setting. The challenge dataset is based on the Open Images V5 dataset [5], which contains 9 million images annotated with class labels, bounding boxes, segmentation masks and visual relationships.

The challenge task is to detect objects and their associated relationships. The relationships include human-object relationships (e.g. "man holding camera"), object-object relationships (e.g. "spoon on table"), and object-attribute relationships (e.g. "handbag is made of leather"). Each of the relationships can be expressed as a triplet, written as a pair of objects connected by a relationship predicate e.g. ("beer", "on", "table"). Visual attributes are also triplets where object is connected with an attribute using the "is" relationship e.g. ("table", "is", "wooden"). The challenge contains 329 unique triplets, which span 57 different object classes, 5 attributes, and 10 predicates. In this paper we present our solution which ranked first out of over 200 teams, and outperformed the second place team by over 5% on the held-out private leaderboard. To make our approach more practical, we focus on minimizing the hardware requirements during training. Specifically, we show that through transfer learning we can significantly speed up optimization, and train the entire model on a *single* GPU in under two days.

2. Our Approach

Our pipeline consists of three stages. In the first stage, object detection model fine-tuned for the challenge classes generates object bounding boxes along with their associated confidences. In the second stage, two separate models based on gradient boosting and convolutional neural networks are used to model spatio-semantic and visual features. Finally, a third stage model takes outputs from the first two stages as input and generates the final prediction. In this section, we describe each stage in detail, and Figure 1 summarizes the entire pipeline.

*Authors contributed equally and order is determined randomly.

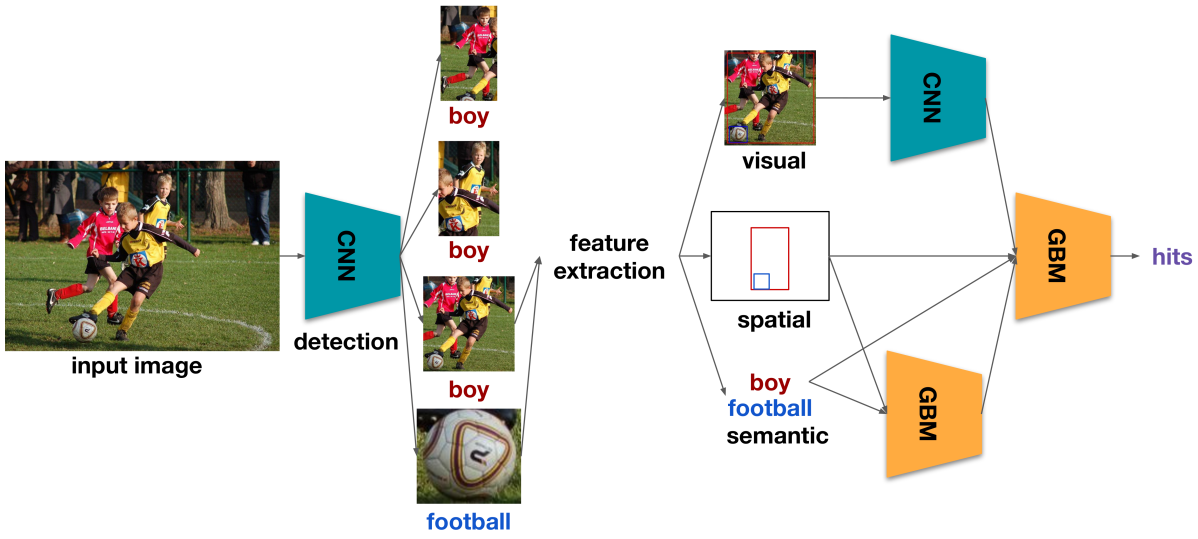


Figure 1: Proposed visual relationship model architecture. Object detection model is first applied to get bounding boxes for every object in the input image. Spatio-semantic and visual models are then applied to candidate object pairs to generate initial relationship predictions. Finally, features and model predictions are combined in the last stage to output the final relationship prediction. In this example our model outputs ("boy", "hits", "football").

2.1. Object Detection With Partial Weight Transfer

Training object detection model from scratch in a reasonable amount of time on a large dataset requires significant resources. For instance, one of the leading models on the Open Images 2018 Object Detection Track needed 33 hours of training on 512 GPUs [1]. Instead of running multiple costly and time consuming training experiments, we focus speeding up optimization with limited hardware resources. In order to achieve this, we propose the partial weight transfer strategy. The main idea is to transfer as much information from models trained on well-established object detection benchmarks such as COCO [6] and Open Images [5]. Minimal fine-tuning is then performed on the target task dataset.

Transfer learning is a popular and economic approach for improving generalization by transferring knowledge between datasets and domains. However, choosing which model parts to keep or discard can have a significant impact on the performance. A common approach in object detection is to use a popular model such as Faster RCNN [7] or Retina Net [9] pre-trained on large and general datasets where high performance is observed. Then the classification and regression heads are replaced with randomly initialized weights to train task-specific detectors. However, we observed that fine-tuning the network this way can still take significant amount of time to converge, and doesn't always achieve high accuracy. To improve convergence we propose to also initialise classification and regression heads with pre-trained weights by approximately matching classes between datasets.

We denote source and target task datasets as $\mathcal{S}_{\text{src}}$ and $\mathcal{S}_{\text{task}}$ respectively. $\mathcal{S}_{\text{src}}$ is typically a large public dataset such as COCO on which many of the leading models are trained and released. $\mathcal{S}_{\text{task}}$ in our case is the challenge dataset, and the aim is to transfer models from $\mathcal{S}_{\text{src}}$ to $\mathcal{S}_{\text{task}}$ with high accuracy and minimal computational resources. The main difficulty here is that the target dataset typically contains classes not present in $\mathcal{S}_{\text{src}}$. However, we hypothesize that there should be common information learned by the model for related classes between the two datasets. Following this intuition, it should be possible to partially transfer model weights for related classes from $\mathcal{S}_{\text{src}}$ to $\mathcal{S}_{\text{task}}$ and improve fine-tuning. Figure 2 demonstrates this process. In this example, weight vectors in the classification head of $\mathcal{S}_{\text{src}}$ model associated with "car" and "dog" classes are directly copied to corresponding classes in $\mathcal{S}_{\text{task}}$. Similarly, weights for the more general class "person" are transferred to related classes "woman", "boy", "girl", and "man" in $\mathcal{S}_{\text{task}}$. Weights for classes that don't have a match are randomly initialized. Formally, given a mapping $k \rightarrow g(k)$ from task class index k to source class index, the classification head layer for the task model has the following structure:

$$z_k = \begin{cases} \omega^{\text{task}}_k \cdot x^n + b^{\text{task}}_k & \text{if } g(k) = \varnothing \\ \omega^{\text{src}}_{g(k)} \cdot x^n + b^{\text{src}}_{g(k)} & \text{otherwise} \end{cases} \quad (1)$$

where x^n is the input from previous layer, $\omega^{\text{src}}_{g(k)}$ and $b^{\text{src}}_{g(k)}$ are weights and biases transferred from the source dataset, and ω^{task}_k and b^{task}_k are randomly initialized parameters.

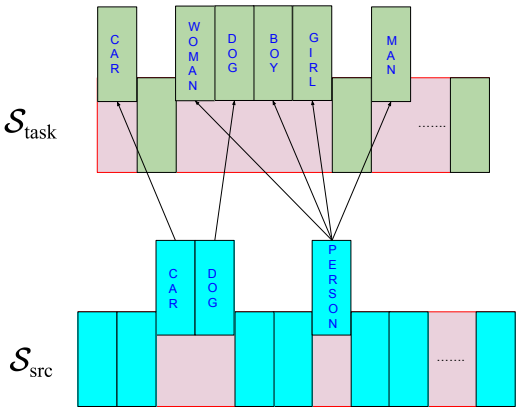


Figure 2: Partial weight transfer diagram example. Classification head weights for "car" and "dog" classes are directly copied to corresponding classes in $\mathcal{S}_{\text{task}}$. Weights for the more general class "person" are extrapolated to related classes "woman", "boy", "girl", and "man" in $\mathcal{S}_{\text{task}}$.

Applying this weight transfer even for approximate class matches such as "person" $\rightarrow$ "boy", enable us to significantly accelerate fine-tuning to less than one day on a single GPU and achieve better accuracy. In the challenge we use detection models pre-trained on the popular COCO dataset which has 80 classes. After matching classes between datasets we were able to transfer classification weights for 44 of the 57 challenge classes.

Fine-tuned detection model can be independently evaluated on the related Open Images 2019 Object Detection challenge by submitting predictions only for classes that are common to both challenges. Table 1 shows detection mAP performance for the Cascade RCNN [2] model pre-trained on COCO and fine-tuned for visual relationship classes. From the table we see that partial weight transfer significantly improves leaderboard performance by over 35%. We also found that training time was considerably reduced from around one week to less than a day. Blending multiple models with test time image augmentation provides additional performance boost, and we use this approach as the first stage in our pipeline.

2.2. Spatio-Semantic and Visual Models

Given the bounding boxes predicted by the object detection model, the relationship model aims to (1) detect whether a two objects are related, and (2) predict their relationship. These tasks require simultaneously learning spatial, semantic and visual features. In our experiments, we find tree-based gradient boosting models (GBMs) to be effective for learning spatial and semantic features, while convolutional neural network (CNN) models excel at capturing visual features. The second stage in our pipeline thus utilizes both GBM and CNN models to perform feature extraction for pairs of objects.

Object Detection Results (mAP)		
Model	Validation	LB
Cascade RCNN [2]	0.43	0.048
Cascade RCNN [2] + PWT	0.53	0.065
Blend + TTA	**0.56**	**0.068**

Table 1: Results on the Open Images 2019 Object Detection challenge. Only 57 classes from the visual relationship challenge are submitted. PWT and TTA denote partial weight transfer and test time image augmentation respectively.

Spatio-Semantic Model. Spatial information such as location of the object in the image and relative position between objects, plays an important role in relationship detection. Objects that are far away from each other less likely to have a relationship, and relative position between objects can be very informative when determining relationships such as "on" or "under". Semantic information on the other hand, can capture the likelihood of the two objects co-occurring together or having a certain type of relationship. We describe both types of information through features and train a GBM model to predict relationship type. The features include:

1. Object spatial features - we use features such as relative and absolute position of the object in the image and size of the object (estimated by its bounding box).

2. Object semantic features - we include features such as other objects that this object typically appears with and types of relationships that in commonly has.

3. Pairwise spatial features - we encode information such as relative position of the two objects, IOU between their bounding boxes and Euclidean distance between box centers.

4. Pairwise semantic features - similar to pairwise spatial features, we summarize how frequently the two objects appear together and types of relationships they typically have.

We consider two alternatives for defining the GBM training objective. The first option is to train a single GBM for multi-class classification over all the possible relationships with an additional "None" class for no relationship. The second option is to train separate GBM models for every relationship type with a binary classification objective. For example, for the relationship type "under", we find all pairs of objects that can possibly form an "under" relationship. Then label those that are present in the ground truth set as

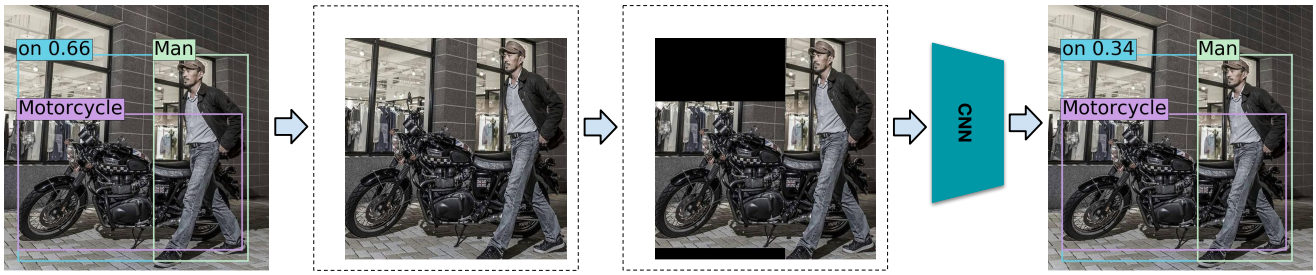


Figure 3: Visual model pipeline example on one the challenge images. Here, spatio-semantic model is unable to correctly predict the relationship and outputs high probability for "on". Cropping the bounding boxes for "man" and "motorcycle", blacking-out the background and passing the resulting image through the visual CNN model reduces the probability for "on" from 0.66 to 0.34.

positive samples and others as negative samples. The advantage of the first option is that it is more computationally efficient and only requires a single model for all types of relationships. However, empirically we find that the second option performs consistently better. We presume that this is due to the fact it allows the model to separately focus on each relationship improving generalisation.

Visual Model. Models that rely solely on spatial and semantic features have failure modes that can only be corrected with visual information. One example of such failure mode is shown in Figure 3. Here, both spatial and semantic features indicate that the likely relationship is ("man", "on", "motorcycle"). However, from visual inspection it is clear that the man is actually *next* to the motorcycle and not on it. To incorporate visual information we use a CNN-based architecture. Given a pair of objects for which we aim to predict the relationship, we first crop the image so it only contains the union of the bounding boxes for the two objects. Then for each pixel in the cropped image that does not belong to the bounding box of either object, we turn it into background by making it black. This reduces background and scene clutter, and enables the model to focus on the target objects. Analogous to the spatio-semantic model, we also find that training separate models (fine-tuned from the same backbone) for each relationship type yields better results than single multi-class model. From Figure 3 we see that the visual model reduces the probability of ("man", "on", "motorcycle") from 0.66 to 0.34.

2.3. Model Aggregation

The last stage takes predictions from the spatio-semantic and visual models, and combines them to make the final prediction. A straightforward way to combine models is through averaging. However, depending on the properties of the input scene, different types of model tend to perform better and need to be selected accordingly. Averaging prevents such specialisation, so in the last stage we train another model that takes as input predictions from the second stage together with image and target object pair features, and learns how to optimally combine them. We also use GBM here as decision trees can learn highly non-smooth decision boundaries that are beneficial for specialisation. To train the model we split the official training set into two parts. All second stage models are trained on the first part, and the ensemble model is trained on the second part.

3. Experiments

The challenge is based on the Open Images V5 dataset which is a large-scale multi-modal collection of over 9 million images. A subset of 1,743,042 images contain bounding boxes, and we use this subset to train and validate the object detection model. The challenge dataset is a subset of the Open Images V5 data, and contains 391,073 labelled relationship triplets from 100,521 images. There are a total of 329 unique triplets with 287 object-object relationships over 57 unique object classes, and 42 object-attribute "is" relationships over 5 attributes. Furthermore, an additional 99,999 images are used as the held-out test set that is split 30%/70% for public and private leaderboards respectively. All test labels are hidden and model evaluation is done by submitting predictions to the Kaggle platform. Public leaderboard score is available throughout the competition, while private leaderboard is released at the end and used to compute final team rankings.

Class Imbalance. The object detection dataset contains significant class imbalance with a long tail. Randomly sampled mini-batch training with skewed class distribution over emphasizes frequent classes. Since challenge objective assigns equal weight to every class, model bias towards popular classes can significantly hurt performance. To address this problem we adopt a sampling strategy that approximately balances class distribution during training. Let n_k denote the number of images containing class k in the train-

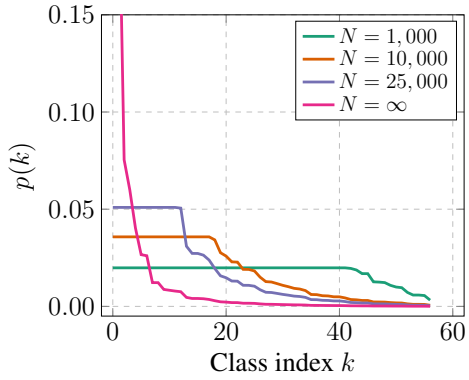


Figure 4: Class probabilities $p(k)$ sorted from highest to lowest for all 57 classes as N in Equation 3 is varied from ∞ (original distribution) to 1,000.

ing set with K classes. Randomly sampling from the training set results in the following probability for each class:

$$p(k) = \frac{n_k}{\sum_{i=1}^{K} n_i} \tag{2}$$

Frequent classes with high image count n_k thus have much higher probability of being included in each mini-batch. To balance the probabilities we introduce an additional parameter N and sample images according to:

$$p(k) = \frac{\min(n_k, N)}{\sum_{i=1}^{K} \min(n_i, N)} \tag{3}$$

The effect of N is shown in Figure 4. The figure shows class probabilities for all 57 classes sorted from highest to lowest as N is varied from ∞ (original distribution) to 1,000. We see a gradual effect where class distribution approaches uniform distribution as N is decreased. Empirically, we found that setting N in the [1,000, 10,000] range produced better performance than using original or uniform distribution. Compared to the original class distribution, the Cascade RCNN model improved in performance from 0.44 to 0.53 on the validation set, and from 0.058 to 0.065 on the object detection leaderboard.

Learning Setup. We split the challenge dataset into 374,768 triplets to train the second stage spatio-semantic and visual models, and 12,314 triplets to train the third stage aggregation model. All models are validated on the remaining 3,991 triplets. To evaluate model performance we use the competition metric [1] which aims to capture both object and relationship detection quality. In all experiments we use the mAP_{rel} component of the metric to validate all models. We found it to correlate well with the overall metric and much faster to compute.

[1] https://storage.googleapis.com/openimages/web/evaluation.html

Rank	Team	Public LB	Private LB
1	Layer6 AI	**0.4638**	**0.4080**
2	tito	0.4407	0.3881
3	Very Random team	0.4289	0.3785
4	[ods.ai] n01z3	0.3984	0.3659
5	Ode to the Goose	0.4016	0.3477

Table 2: Final team rankings on the public and private leaderboards.

3.1. Implementation Details

Our pipeline consists of three stages that include object detection, spatio-semantic and visual information extraction, and final aggregation. In this section we describe the implementation details for each stage.

Object Detection. For the object detection stage, we use an ensemble of Cascade RCNN [8] detection networks with ResNeXt [11] and HRNet [10] backbones trained on COCO and fine-tuned using our partial weight transfer approach. As described in Section 2.1, we are able to initialise 44 out of the 57 challenge classes by mapping them onto the 80 COCO classes. The other 13 classes are either initialized randomly or transferred from the backbone that is fine-tuned for the challenge classes without partial weight transfer. To combine multiple detection models we use a weighted non-maximum suppression (NMS) approach where bounding boxes from all detection models are combined using a weighted average. The weights for each model are selected according to performance on the validation set. This approach is similar to traditional NMS except instead of choosing the most confident box we use weighted average. Empirically, we found that weighted average provided a gain of around 2 points on the leaderboard.

For training of visual relationship models, we use ground truth bounding boxes instead of predicted ones. We also experiment with using the predicted boxes which we expect to perform better. The rationale is that exposing the relationship model to errors (e.g. shifted bounding boxes or mislabeled classes) made by object detection should enable it to learn to correct them and make more robust predictions. However, this does not perform well, and we presume that this is because the relationship model is not able to sufficiently correct errors made earlier in the pipeline.

Special "is" Relationship. We use a separate pipeline for the "is" relationship since, unlike other relationships, it doesn't operate on pairs of objects. We leverage the object detection model and modify the classification head to predict over all object-attribute pairs that can form a valid "is" relationship as separate classes. One concern here is that there aren't enough training examples to learn a reliable detection model for each pair. We address this problem by again leveraging the partial weight transfer approach.

Relationship	Spatio-Semantic	Visual	Avg.	3'rd Stage
plays	0.49	0.58	0.55	**0.59**
hits	0.58	0.47	0.58	**0.61**
at	0.37	0.35	0.35	**0.42**
inside_of	0.31	0.35	0.32	**0.37**
interacts_with	0.42	0.42	0.41	**0.44**

Table 3: Validation AP_{rel} results for a subset of five relationships. We show performance for spatio-semantic and visual models, and two ways of combining them using weighted average (Avg.) and 3'rd stage GBM model.

Attribute	AP_{rel}	Best Class	Best AP_{rel}	Worst Class	Worst AP_{rel}
transparent	0.40	bottle	0.75	table	0.13
wooden	0.60	guitar	0.95	bench	0.10
plastic	0.39	piano	0.68	bench	0.02
leather	0.46	sofa	0.68	suitcase	0.25
textile	0.62	sofa	0.80	suitcase	0.40

Table 4: Breakdown of the "is" model results by attribute. For each attribute we show validation AP_{rel} results together with best and worst performing class.

This time we transfer weights from one of our base detection models and then fine-tune on the available "is" relationship training data. For instance, both "wooden" and "plastic" piano classes get initialized with the piano classifier weights from our base detection model as well as its backbone. Fine-tuning this way makes the model more robust to lack of training data and improves performance.

Spatio-Semantic and Visual Models. For spatio-semantic model we use the tree-based GBM architecture from the XGBoost library [3] due to its excellent performance in our experiments. We train a separate model for each relationship type by framing the problem into binary classification. Specifically, we iterate over all ground truth object bounding boxes and for each pair of objects that can form the target relationship we check whether that pair is in the ground truth training relationship set. If it is, we label it as a positive sample, and if it is not as a negative sample. Note that negative samples are approximate here since two objects can have a relationship that is not labelled in the training set. However, the probability of that is small and empirically we found that using this procedure with negative samples produced good performance. We use the same set of hyper-parameters for all relationships, the GBM model is trained with the `dart` booster and `max_depth` set to 10. To prevent over-fitting, we further set `subsample` and `colsample_bytree` parameters to 0.2, as well as `gamma` and `lambda` parameters to 2.0 and 1000 respectively. Each model is trained for 5000 boosting iterations with an early stopping check every 50 iterations.

For the visual model we use the ResNeXt backbone from the object detection model that has been fine-tuned for the challenge classes. We apply a 3-layer MLP on top of the backbone with ReLU activations. The last layer outputs a binary sigmoid prediction, and we train this model using the same positive/negative samples as the spatio-semantic model. For all relationships, we use the same batch size of 32 and run optimization for 35 epochs. To reduce overfitting, we apply dropout with $p = 0.2$ to each MLP layer. We use the Adam optimizer with cosine learning rate annealing and linear learning rate warmup. The maximum and the minimum learning rates are $3\mathrm{e}^{-4}$ and $5\mathrm{e}^{-5}$ respectively.

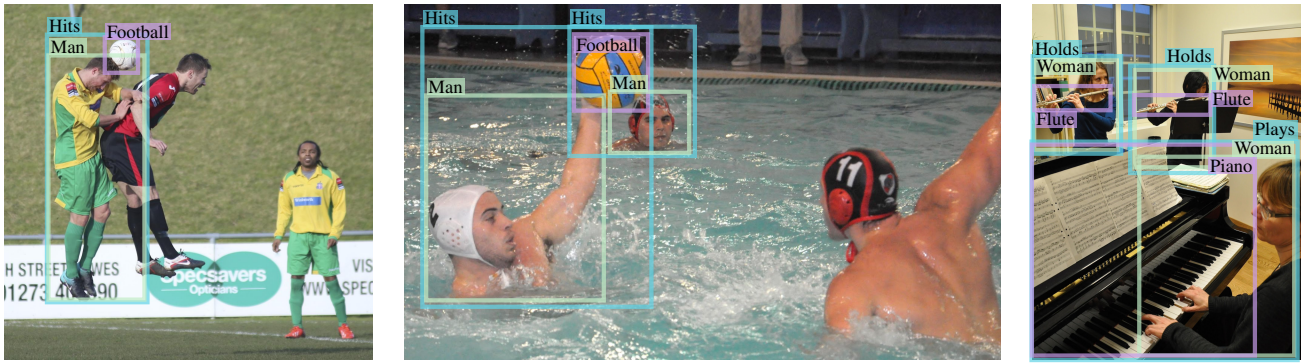
Final Aggregation. In the last stage we combine predictions from spatio-semantic and visual models together with object features (see Section 2.2) to generate the final relationship prediction. We also use a tree-based GBM model here, and train it with the XGBoost library. The parameters for this model are the same as for the spatio-semantic model, with the only difference that we use `gbtree` booster instead of `dart` and lower tree depth to 8.

3.2. Results

The final team standings are shown in Table 2. Our team "Layer6 AI" outperforms all other teams on both public and private leaderboards beating the second place team by over 5%. These results indicate that our multi-stage pipeline is highly robust and produces leading performance on this challenging task. We can also conclude that by applying transfer learning through our partial weight transfer approach we can train highly accurate visual relationship models with minimal hardware requirements.

Table 3 summarizes performance for each stage across a sample of five relationships. We see that the spatio-semantic model performs better on "at" and "hits" relationships, while visual model outperforms on "plays" and "inside_of". This validates our hypothesis that both types of information are required to accurately detect all relationships. We also see that combining these models through averaging is highly sub-optimal and actually hurts performance for four out of the five relationships. This observation motivates us to introduce a third stage to learn how to better combine spatio-semantic and visual predictions. Results for the third stage are shown on the right in Table 3, we see that the third stage model is able to effectively learn when to use each type of signal and further improve performance. We are able to consistently improve performance over the best individual model on all five relationships, with particularly significant improvement on the "at" relationship where we gain over 5 points in mAP_{rel} or 11%.

We described in Section 3.1 that the "is" relationship is



(a) (b) (c)

Figure 5: Qualitative results from our model. Green and purple bounding boxes are detected subject and object classes, and purple box is the predicted relationship between them.

treated differently in our pipeline since instead of pairs it operates on object-attribute combinations. Table 4 shows the AP_{rel} performance of the "is" model for each of the five attributes. For each attribute we also show best and worst object class with corresponding AP_{rel}. From the table we see that that the model performs well on attributes wooden and textile, and does significantly worse on transparent and plastic. As expected performance is highly dependent on the number of training instances for each object-attribute pair, as well as label ambiguity. For instance, many plastic objects such as bottles are also labelled as transparent, and the model has difficulty distinguishing between the two properties. By analysing the best and worst class for each attribute we can directly observe the effect of the training set size. Textile sofas and suitcase appear much more frequently in the training data (and arguably in real life) than leather ones so performance on textile is better. Interestingly, the model has difficulty recognizing attributes such as transparent, wooden and plastic for common furniture items such as tables and benches. After inspecting the data we observed that furniture objects have a lot of variably, and often appear in cluttered scenes with many occlusions making it challenging to identify what they are made off.

Qualitative examples are shown in Figure 5. Figure 5a shows a difficult scene where two soccer players are trying to get the ball but only one of them hits it. Our model is able to correctly identify which player hit the ball which indicates a degree of robustness to spatially complex scenes. Figure 5b shows a related failure case where player hits the ball during a game of water polo. The model is able to correctly capture that relationships, but also identifies another player as hitting the same ball which is incorrect. Possible reasons for this failure can be partial occlusion between the two players, and position of the incorrectly identified player relative to the ball. Position in particular is difficult to capture accurately here, the two bounding boxes are close together in 2D but the player is actually far from the ball in 3D. 3D spatial information is difficult to capture with a single image and we hypothesise that performance can be improved if another view or depth information is added as input. Finally, Figure 5c shows a more cluttered scene where multiple musicians are playing various instruments such as flute and piano. Here, we see that our model is able to correctly identify all relationships even though musicians are in close proximity to each other.

4. Conclusion

We present our winning solution to the Open Images 2019 Visual Relationship challenge. We propose a novel partial weight transfer approach to effectively transfer learned models between datasets and accelerate training. Our pipeline consists of object detection followed by spatio-semantic and visual feature extraction, and a final aggregation phase where all information is combined to generate relationship prediction. Partial weight transfer enables us to train the entire architecture in under two days on a *single* GPU making it accessible to most researchers and practitioners. In addition to high efficiency, we also achieve top performance and beat over 200 teams to place first in the competition outperforming the second place team by over 5%. In the future work we aim to focus on fusing the three stages into a joint architecture that can be trained end-to-end. We hypothesize that end-to-end training can improve the flow of information and lead to better performance.

References

[1] Takuya Akiba, Tommi Kerola, Yusuke Niitani, Toru Ogawa, Shotaro Sano, and Shuji Suzuki. Pfdet: 2nd place solution to open images challenge 2018 object detection track. *arXiv preprint arXiv:1809.00778*, 2018.

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, 2018.

[3] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD*, KDD '16, New York, NY, USA, 2016. ACM.

[4] Vincent Chen, Paroma Varma, Ranjay Krishna, Michael Bernstein, Christopher Re, and Li Fei-Fei. Scene graph prediction with limited labels. In *ICCV*, 2019.

[5] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.

[6] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[8] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.

[9] Ross Girshick Kaiming He Tsung-Yi Lin, Priya Goyal and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[10] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *CoRR*, abs/1908.07919, 2019.

[11] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[12] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *ECCV*, pages 670–685, 2018.

[13] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *CVPR*, 2018.

[14] Ji Zhang, Kevin J. Shih, Ahmed Elgammal, Andrew Tao, and Bryan Catanzaro. Graphical contrastive losses for scene graph parsing. In *CVPR*, 2019.