November 29, 2020

Using Machine Learning for Object Detection in StarCraft

Aeon Williams & Alejandro Herrera

Abstract

StarCraft is a video game that has been around for as long as Aeon has been alive, and has become a well known iconic piece of popular gaming culture – the sequel, *StarCraft II*, has become even more important due to it's presence in international e-sports. Because of their complexity, both games have been the subject of many professional machine learning and AI research studies. Most notably, Deep Mind developed AlphaStar, which was the first artificial intelligence to defeat a top professional player in *StarCraft II*. This project was incredibly inspiring, both because of our personal interest in the game series and the incredible machine learning technology advancement it demonstrates. We decided to do object detection in *StarCraft* both as a challenging project for ourselves, and to contribute to the same niche as AlphaStar, just within our personal skillset.

There are many examples of algorithms such as Fast-RCNN, Mask-RCNN, YOLOv3, etc… that implement multiclass object detection of multiple objects in an image, mostly as pre-built professionally made packages. Although RCNN is not technically as advanced or efficient as it's successors, we saw merit in having a simple implementation of this algorithm, if only to compare to other "better" ones. Given our current experience as students, this project filled the requirement of being something that a) has not been done a million times and b) pushes our knowledge base without being too far out of scope.

We use Keras and Tensorflow to build a region based *Convolutional Neural Network* (CNN) to perform multiclass object detection of multiple objects in images, using methods outlined by Ross Girshick. For each input image, *Selective Search* is used to create approximately two thousand proposed regions where it thinks an object may be. This region proposal is fed into the *Convolutional Neural Network*, which predicts a class label and the probability it's accurate, and *Non-maximum Suppression* is used to tighten bounding boxes for each identified object whose probability is above a threshold of 0.98.

The algorithm tested with high accuracy results but is ultimately extremely slow and produces many false positives. Our finished product does not make trustworthy predictions and will not achieve our goal of using model-built army composition to predict the winner of a match, at least without heavy adjustments and further research.

Introduction

StarCraft is a Real Time Strategy (RTS) game by Blizzard Entertainment that has become a cultural phenomenon so significant, it has worldwide popularity comparable to a country's national sport. Like a national sport, the ability to predict winners of *StarCraft* matches is extremely valuable, and a very complex undertaking. There are many variables to consider when attempting to accurately predict a winner, but the one we are focusing on is army composition. Army composition of each player must be carefully examined and evaluated against each other, accompanied by predetermined knowledge of which units are statistically better compared to the other player's. This project uses object detection in images to compile army composition of both players at the same timestamp during a match, to then evaluate how "good" each army is and what their current chances of winning are.

November 29, 2020

Literature Review

Paper 1 pm CNN based stock market prediction is an interesting paper because it leans on one of the strengths of CNNs, which is letting the algorithm by itself determine which features are important by expanding the amount of available features from sources others didn't consider important at the time (in this case, data from different markets). Including this data resulted in better predictions of the next day's stock market movements. Paper 5 on Rich feature Hierarchies touched on this strength also while acknowledging that labeled training data can be scarce, and proposing scalable solutions for training via supervised pre-training and domain-specific fine-tuning. Paper 3 on Classification for Artery Stenosis also addresses also mentions lack of large datasets, as well as the limitations of handcrafted features requiring manual preprocessing, making them less scalable. Most papers do touch on lack of data being a limiting factor, but others such as Paper 2 on CSPNet also mention the limits imposed by computing power and supporting networks as well as methods on how to address these limitations. Paper 4, Mark R-CNN, was more of a presentation of the general framework for object instance segmentation, and most closely resembled our work as it was a straightforward implementation of R-CNN with small optimization added to increase speed. All papers agree that R-CNN is a highly robust and exceptionally good tool to tackle the issues of object instantiation and classification.

Datasets

A. Raw Training Dataset Creation

StarCraft has two possible art styles: “cartooned” and “normal”. We originally planned on using the “cartooned” style, but quickly realized that the lack of pixel variation lead to unavoidable incorrect classifications. The “normal” art style was chosen instead. Images include five units each of Zerg and Terran:

Mutalisk, Ultralisk, Hydralisk, Scourge, Zergling, Firebat, Tank, Valkyrie, Marine, Goliath

The dataset is composed of image screenshots of the entire *StarCraft* screen during a match. These images have a variable number and variety of units. Image resolution is consistent, meaning that all units maintain relative size compared to each other throughout all screenshots.

Figure 1: A screenshot of StarCraft, *from the perspective of the Zerg player.*

The training dataset contains 50-75 unique screenshots saved as png files. Vott was used to quickly generate bounding box and class label information of each object in the collected screenshots.

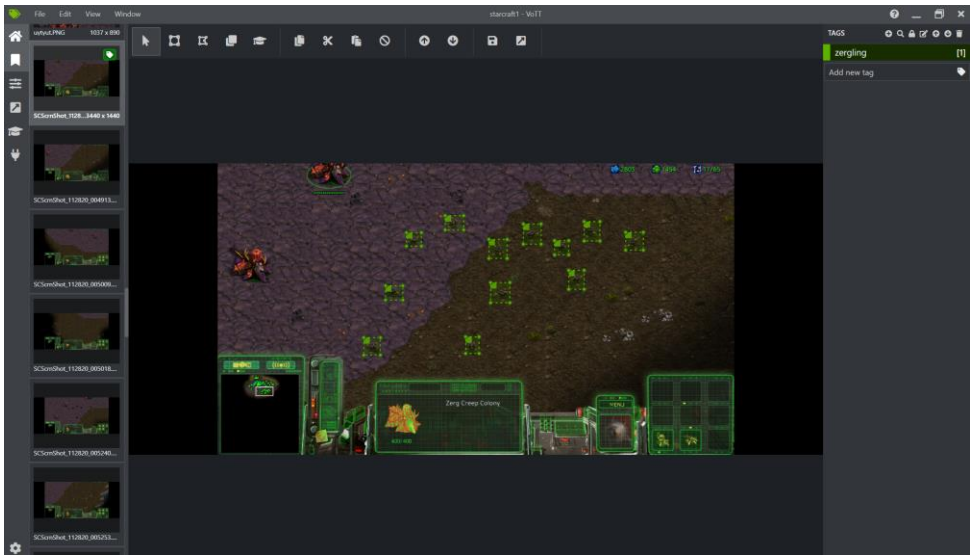


Figure 2: Using Vott to define and label objects in images.

The image information was exported as a csv, which was then converted to an xml file containing all information about the dataset. We wrote a tool to segment this master xml file into unique xml files corresponding to each unique image. Snippet of xml file for "SCScrnShot_112820_004631.png":

```
<root>
<row>
  <image>SCScrnShot_112820_004631.png</image>

  <xmin>1256.8075445337058</xmin>
  <ymin>592.869365609349</ymin>
  <xmax>1367.3489346838978</xmax>
  <ymax>680.6156093489149</ymax>
  <label>zergling</label>
 </row>
```

```
<row>
  <image>SCScrnShot_112820_004631.png</image>
  <xmin>1358.9381767376876</xmin>
  <ymin>336.84265442404006</ymin>
  <xmax>1463.4718826405867</xmax>
  <ymax>419.78088480801335</ymax>
  <label>zergling</label>
</row>
```

The finished dataset has two parts: screenshot images, and corresponding xml files with details about objects in those images.

B. Dataset Preprocessing

Taking screenshots and defining/labeling objects are the only dataset handling processes that were not done with code. For each screenshot, *Selective Search* is used to segment possible areas of interest into 128x128 sized images. These segments are compared against the known bounding box locations, and if the *Intersection over Union* (iou) is above a threshold of 0.7, this segment is labeled properly and stored in a "positive" directory – if the iou is lower than the threshold, it is labeled and stored in a "negative" directory. This results in a training set of consistently sized images that is greater than seven times the size of the raw dataset, because *Selective Search* identifies known objects a variable number of times.

Additional dataset size bloating is performed before training the model. Each existing image segment gets copied and augmented, to increase variety within the dataset. Because *StarCraft* units only have a few orientations, we decided to only use horizontal flipping augmentation for this project.

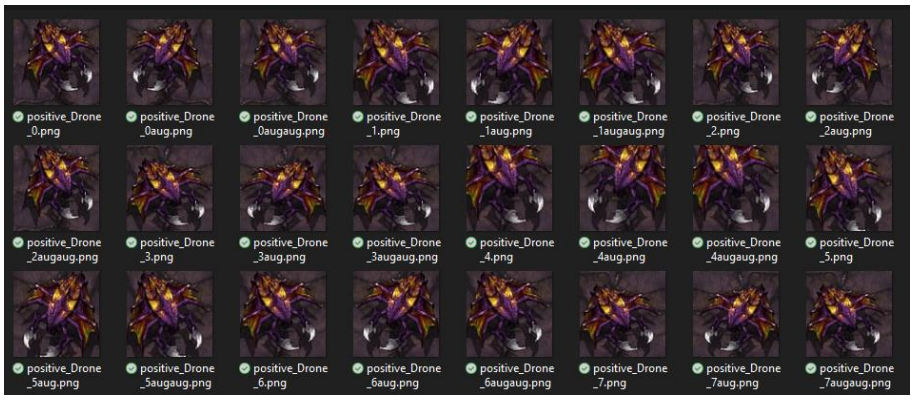


Figure 3: Snippet of the "positive" directory when all dataset handling processes are complete. This is the final dataset for training the model.

Any test images that get fed into the model have a near identical preprocessing methods. *Selective Search* is used to segment the image into proposed 128x128 regions, where it thinks there may be objects to classify. These segments are fed into the model for class label predictions.

C. Additional Dataset

Although it seemed logical to keep all train and test images at the same resolution, we took additional screenshots with variable resolution for evaluating differences in results between the two methods. These images were gathered and preprocessed exactly like the steps mentioned above.

Methods

A. Selective Search

Selective Search is a region proposal algorithm. It starts by segmenting an image using the method outlined by Ferzenszwalb and Huttenlocher.

This list of segments is used to:

1. Add all bounding boxes corresponding to segmented parts to the list of region proposals
2. Group adjacent segments in the region proposal list based on similarity
3. Repeat, creating larger and larger segments that are added to the list of region proposals.

B. Intersection over Union

Intersection over Union is an evaluation metric for accuracy of object detection algorithms. It takes the ground truth bounding boxes (hand labeled) and predicted bounding boxes and finds the difference between their Area of Overlap & Area of Union. An iou of one means the boxes have identical dimensions, and 0 means the boxes are very far apart.

C. Convolutional Neural Network

A sequential model is built with individually specified layers:

Layer 1: Two-dimensional convolution layer with Rectified Linear Activation, 3x3 filter matrix size, 32 nodes, and input with shape 128x128x3, with 3 signifying the images are RGB.

Layers 2-10: Alternating convolution and max pool layers, with the same stats as layer 1 sans input shape

Layer 11: Flattens network to connect convolution and dense layers

Layers 12-14: Multiple dense layers to increase complexity of model's mathematical functions

Layer 15: Final dense layer, with the number of class labels we can predict, and softmax to generate probabilities to sift through

D. Non-maximum Suppression

Non-maximum Suppression is an algorithm used to filter out region proposals based on overlap threshold. The algorithm starts with a full “active” list, and empty “final” list, and completes the process of:

- Selecting the proposals with the highest confidence score, remove it from “active” and add it to “final”. Remove proposals from “active” whose *Intersection over Union* with this proposal is above the threshold 0.1. Repeat until “active” is empty.

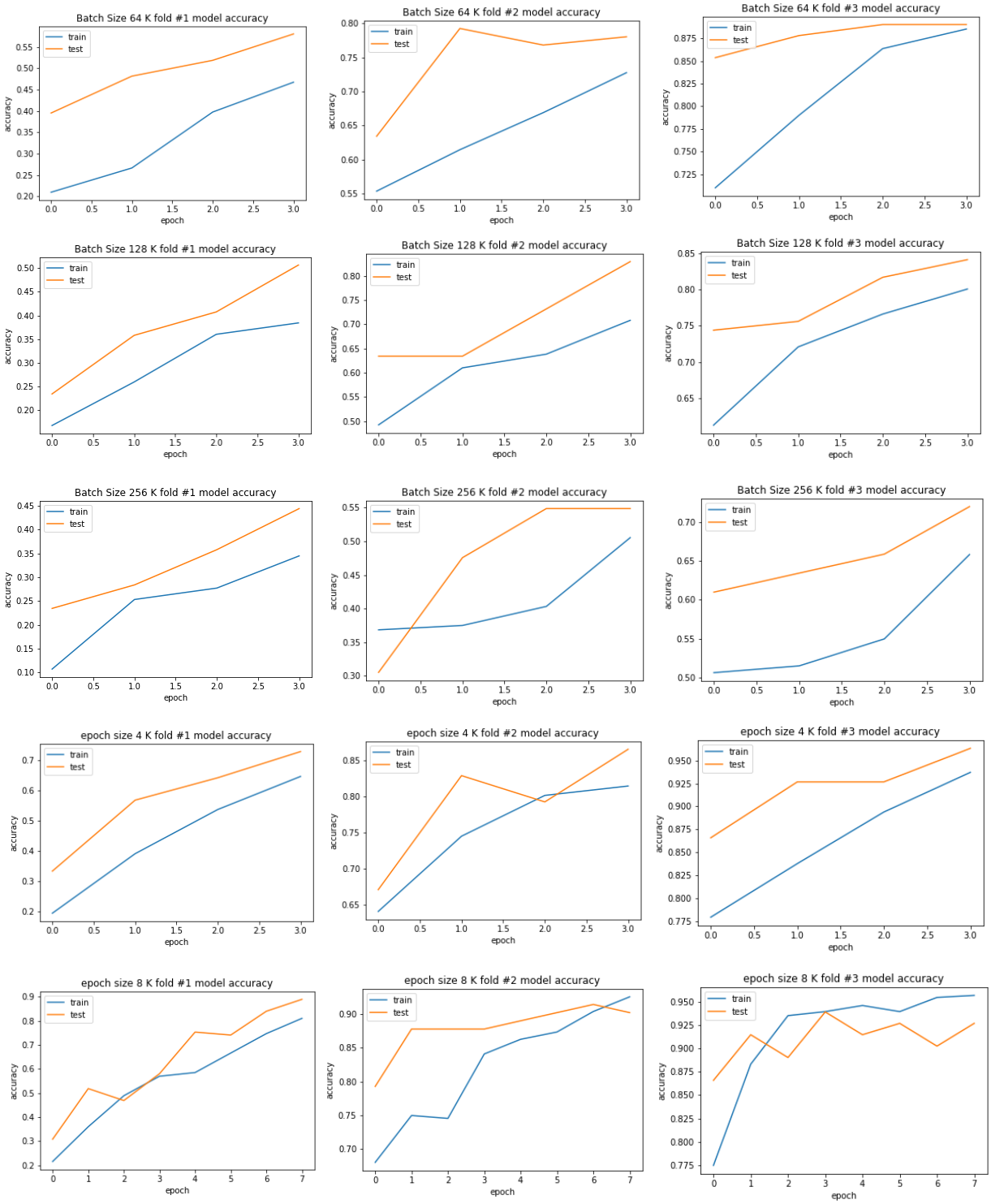
November 29, 2020

Algorithm Analysis

A. Result Evaluation

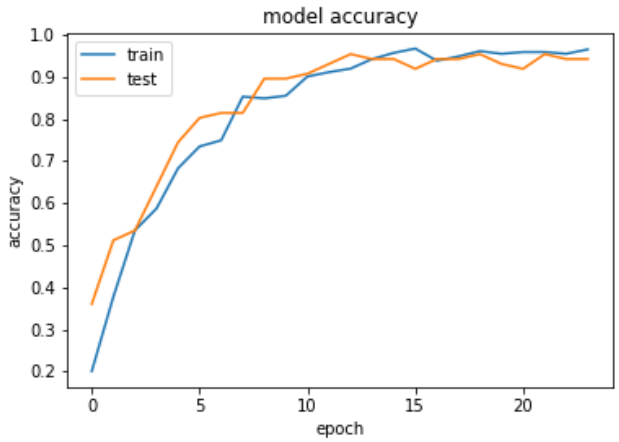
Selective Search is very slow with the resolution of our image samples. This is consistent with the researched and documented issues of region based *Convolutional Neural Network*s, which is why more efficient algorithms have since become the standard. With the current screenshot resolution, it takes approximately ten minutes to define and predict all objects of a test sample. This model is too slow to work in real time and fails to meet our proposed practical application.

Several parameters were evaluated with a range of values to compare results.



November 29, 2020

The model had overall comparable skill on train and test for all parameter tests. An increased number of epochs yielded better results, with the final model being trained with early stopping measures, at epoch 24. Batch size had a lot of variation, and ultimately 64 was chosen as the final value for the model. The final model yielded great results statistically:



But, has many false positives and mislabels:

```
{'negative': 0,
 'mutalisk': 1,
 'ultralisk': 2,
 'firebat': 3,
 'tank': 4,
 'hydralisk': 5,
 'valkyrie': 6,
 'marine': 7,
 'scourge': 8,
 'goliath': 9,
 'zergling': 10}
```

```
pred: 0
pred: 1
pred: 6
pred: 7
pred: 5
pred: 10
pred: 3
pred: 2
pred: 9
pred: 4
pred: 8
```


```
pred: 0
pred: 6
pred: 3
pred: 7
pred: 5
pred: 2
pred: 1
pred: 4
pred: 10
pred: 9
pred: 8
```

B. Areas of Improvement

Deep learning is a very complex undertaking with many variables and algorithms to take into consideration. Our deliverable was on a very strict timeline, so a lot of our evaluations were rushed compared to what they could have been. Deeper evaluation of variables such as:

Bath size, epoch count, cross validation fold count, learning rate, test/train split, validation split, image resolution, dataset size, epoch patience, early stopping, class label count, etc…

could be much more thoroughly tested in a large variety of combinations to determine the absolute best values to get maximum accuracy and minimized overfitting of the model.

Conclusion

Region based *Convolutional Neural Network*s are only practical for very specific cases, and their efficiency should be greatly taken into consideration when contemplating using this algorithm. Our finished implementation is far from being "actually" accurate, so is effectively useless for accurate predictions and army composition building. The dataset is missing most units in the game, yet still fails at being anywhere close to accurate. Major adjustments need to be made to this model for it to be anywhere close to viable, but even then, it will still be unusably slow at making predictions for the proposed use case.

References

A. Python Libraries

- Fernando Pérez and Brian E. Granger. IPython: A System for Interactive Scientific Computing, Computing in Science & Engineering, 9, 21-29 (2007), DOI:10.1109/MCSE.2007.53 (publisher link)
- John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55 (publisher link)
- Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010) (publisher link)
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay**.** Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830 (2011) (publisher link)
- Chollet, F., & others. (2015). Keras. https://keras.io
- Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, & Xiaoqiang Zheng (2016). TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (pp. 265–283).
- Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
- Michael Waskom, Olga Botvinnik, Drew O'Kane, Paul Hobson, Saulius Lukauskas, David C Gemperline, … Adel Qalieh. (2017, September 3). mwaskom/seaborn: v0.8.1 (September 2017) (Version v0.8.1). Zenodo. http://doi.org/10.5281/zenodo.883859
- Clark, A. (2015). *Pillow (PIL Fork) Documentation*. readthedocs. Retrieved from https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf

B. Other

- Ross B. Girshick and Jeff Donahue and Trevor Darrell and Jitendra Malik (2013). Rich feature hierarchies for accurate object detection and semantic segmentation*CoRR, abs/1311.2524*.

- “AlphaStar: Mastering the Real-Time Strategy Game StarCraft II,” *Deepmind*. [Online]. Available: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii. [Accessed: 30-Nov-2020].
- [1] Hoseinzade, Ehsan & Haratizadeh, Saman. "CNNpred: CNN-based stock market prediction using a diverse set of variables. Expert Systems with Applications." (2019). [Online]. Available: https://www.researchgate.net/publication/331911968_CNNpred_CNN-based_stock_market_prediction_using_a_diverse_set_of_variables
- [2] Chien-Yao Wang (IIS Sinica), Hong-Yuan Mark Liao (IIS Sinica), I-Hau Yeh (Elan Microelectronics), Yueh-Hua Wu (IIS Sinica), Ping-Yang Chen (NCTU), Jun-Wei Hsieh (NCTU). "CSPNET: A NEW BACKBONE THAT CAN ENHANCE LEARNING CAPABILITY OF CNN", (2019). [Online]. Available: https://arxiv.org/pdf/1911.11929v1.pdf
- [3] Mariia Dobko, Bohdan Petryshak, and Oles Dobosevych (Ukrainian Catholic University). "CNN-CASS: CNN for Classification of Coronary Artery Stenosis Score in MPR Images.", (2020). [Online]. Available: https://www.researchgate.net/publication/338789914_CNN-CASS_CNN_for_Classification_of_Coronary_Artery_Stenosis_Score_in_MPR_Images
- [4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick (Facebook AI Research). "Mask R-CNN" (2018). [Online]. Available: https://arxiv.org/pdf/1703.06870v3.pdf
- [5] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." (2014). [Online]. Available: https://arxiv.org/pdf/1311.2524.pdf
- International Journal of Computer Vision. September 2004 https://doi.org/10.1023/B:VISI.0000022288.19776.77