EasyChair Preprint

№ 4077

Artificial Superintelligence : An AI That Makes Better AI's Recursively.

Poondru Prithvinath Reddy

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

August 25, 2020

Artificial Superintelligence : An AI That Makes Better AI’s Recursively.

Poondru Prithvinath Reddy

ABSTRACT

Artificial Super Intelligence or ASI that is more potent and refined than human’s intelligence. ASI is based on the ideas that machines can imitate the human mind, their way of working to the extent that they can even supersede them. As a first step, ASI aims to improve the intelligent abilities of the machines and to achieve this, the ASI will have to make an AI which makes better AI’s recursively for achieving high-level intelligence. In this paper, we discuss classic R-CNN model with different SVM architectures/algorithms for object detection implementation to get varying outcomes which in turn results in better AI’s with varying degree of accuracy recursively. However, for simplicity we have taken CNN-SVM combination where an AI algorithm(CNN) passes intelligence to fast machine learning(SVM) algorithm recursively for solving multiclass problems from large data sets that implements object detection for designing a better AI machine. The test results are encouraging with high accuracy and the model is therefore shown that a lesser AI is making a better AI when combined with intelligent vector algorithm recursively resulting in very high level of intelligence.

INTRODUCTION

"**Superintelligence**" refers to the idea that steady advances in artificial intelligence, or machine (computer) intelligence, might one day result in creating a machine vastly superior to humans in reasoning and decision-making abilities.

Artificial Super Intelligence or ASI that has the capability to perform the tasks that are impossible for the human mind to think or do. It is that aspect of intelligence that is more potent and refined than a human’s intelligence. Superintelligence is capable of outperforming human intelligence; it is extremely powerful in doing that. The human brain is made of neurons and is limited to some billion neurons. Superintelligence, therefore challenges this trait, which knows no limit.

The road to endless possibilities of Artificial Super Intelligence is paved by the ideas that machines can imitate the human mind, their way of working to the extent that shortly they can even supersede them. Under these circumstances, it is inevitable that ASI will be much better in concluding tasks that humankind would fail to achieve, and will function in better ways compared to the human.

Object Recognition

Object recognition is a general term to describe a collection of related computer vision tasks that involve identifying objects in digital photographs.

Image classification involves predicting the class of one object in an image. *Object localization* refers to identifying the location of one or more objects in an image and drawing abounding box around their extent. *Object detection* combines these two tasks and localizes and classifies one or more objects in an image.

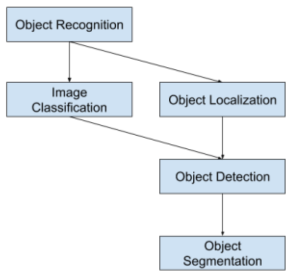
When a user or practitioner refers to "*object recognition*", they often mean "*object detection*".

As such, we can distinguish between these three computer vision tasks:

- **Image Classification**: Predict the type or class of an object in an image.
  - *Input*: An image with a single object, such as a photograph.
  - *Output*: A class label (e.g. one or more integers that are mapped to class labels).
- **Object Localization**: Locate the presence of objects in an image and indicate their location with a bounding box.
  - *Input*: An image with one or more objects, such as a photograph.
  - *Output*: One or more bounding boxes (e.g. defined by a point, width, and height).
- **Object Detection**: Locate the presence of objects with a bounding box and types or classes of the located objects in an image.
  - *Input*: An image with one or more objects, such as a photograph.
  - *Output*: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.

One further extension to this breakdown of computer vision tasks is *object segmentation*, also called "object instance segmentation" or "semantic segmentation," where instances of recognized objects are indicated by highlighting the specific pixels of the object instead of a coarse bounding box.

From this breakdown, we can see that object recognition refers to a suite of challenging computer vision tasks.



Overview of Object Recognition Computer Vision Tasks.

Support Vector Machine(SVM)

Linear SVM is the newest extremely fast machine learning (data mining) algorithm for solving multiclass classification problems from ultra large data sets that implements an original proprietary version of a cutting plane algorithm for designing a **linear** support vector machine.

In machine learning, **Support Vector Machine** (**SVM**) is a **non-**probabilistic, **linear**, binary classifier used for classifying data by learning a hyperplane separating the data. Classifying a **non-linearly** separable dataset using a **SVM** – a linear classifier: However, it can be used for classifying a non-linear dataset also.

Linear classifier (SVM) is used when number of features are very high, e.g., document **classification**. This is because **Linear** SVM gives almost similar accuracy as **non linear** SVM but **Linear** SVM is very very fast in such cases and **non-linear classifier** is useful when data is not linearly separable.

METHODOLOGY

SUPERINTELLIGENCE

However, for AI to lead to superintelligence, we don't necessarily need to develop superintelligent AI ourselves and we only need to develop an AI that is marginally better at developing AI's than we are.

When that happens, we have an AI that makes better AI's that make better AI's — all in the blink of a human eye.

This is known as **recursive self-improvement**, and when AI's master this, superintelligence might be just around the corner.

Now that we are familiar with the problem of object localization and detection, let’s take a look at one top-performing deep learning models.

R-CNN Model Family

The R-CNN family of methods refers to the R-CNN, which may stand for “*Regions with CNN Features*” or “*Region-Based Convolutional Neural Network*,” developed by Ross Girshick, et al.

This includes the techniques R-CNN, Fast R-CNN, and Faster-RCNN designed and demonstrated for object localization and object recognition.

Let’s take a closer look at the highlights of R-CNN technique.

R-CNN

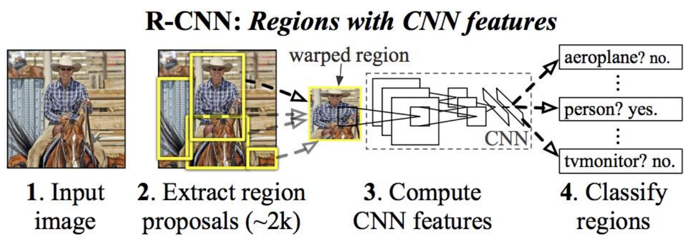
The R-CNN was described in the 2014 paper by Ross Girshick, et al. from UC Berkeley.

It may have been one of the first large and successful application of convolutional neural networks to the problem of object localization, detection, and segmentation. The approach was demonstrated on benchmark datasets, achieving then state-of-the-art results on the VOC-2012 dataset and the 200-class ILSVRC-2013 object detection dataset.

Their proposed R-CNN model is comprised of three modules; they are:

- **Module 1: Region Proposal**. Generate and extract category independent region proposals, e.g. candidate bounding boxes.
- **Module 2: Feature Extractor**. Extract feature from each candidate region, e.g. using a deep convolutional neural network.
- **Module 3: Classifier**. Classify features as one of the known class, e.g. linear SVM classifier model.

The architecture of the model is summarized in the image below, taken from the paper.



The feature extractor used by the model was the AlexNet deep CNN. The output of the CNN was a 4,096 element vector that describes the contents of the image that is fed to a linear SVM for classification, specifically one SVM is trained for each known class.

It is a relatively simple and straightforward application of CNNs to the problem of object localization and recognition. A downside of the approach is that it is slow, requiring a CNN-based feature extraction pass on each of the candidate regions generated by the region proposal algorithm. This is a problem as the paper describes the model operating upon approximately 2,000 proposed regions per image at test-time.

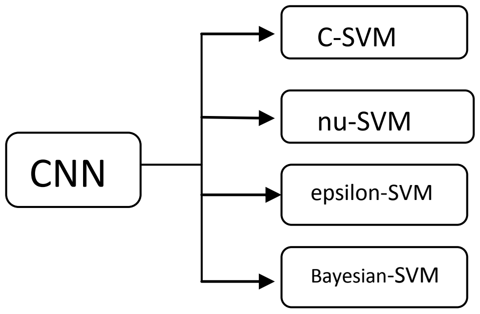
SVM

What are the types of SVM?

According to the form of the error function, SVM models can be classified into four distinct groups: Classification SVM Type 1 (also known as C-SVM classification); Classification SVM Type 2 (also known as nu-SVM classification); Regression SVM Type 1 (also known as epsilon-SVM regression); Bayesian SVM(also known as variational inference (VI) Bayesian SVM).

The methodology is to find the **optimal program following RSI(Recursive Self-Improvement)** procedure defined by given scores and program generation probabilities using Markov chain.

We define following structure of R-CNN model for applying RSI procedure to achieve optimal method of object localization and recognition.



OPTIMAL PROGRAM FOLLOWING RSI

Recursive Self Improvement : Define an improving sequence with respect to G as an infinite sequence of programs P1, P2, P3,... such that for all i > 0, Pi+1 improves on Pi with respect to goal G and G be the identity goal.

Definition: P1 is a recursively self improving (RSI) program with respect to G if and only if Pi(-1) = Pi+1 for all i > 0 and the sequence Pi, i = 1, 2, 3...is an improving sequence with respect to G.

Definition (RSI system).Given a finite set of programs P and a score function S over P. Initialize p from P to be the system's current program. Repeat until certain criterion satisfied, generate $p^{'}$ ∈ P using p. If $p^{'}$ is better than p according to S, replace p by $p^{'}$.

From this definition, one needs to decide how p ∈ P generates a program. In general, we should allow the RSI system to generate programs based on the history of the entire process. The way a program generates a new program is independent, and each program defines a fixed probabilistic distribution over P. This procedure defines a homogeneous Markov chain. We will see that even with this restriction, with some score function, the model is able to achieve a desirable performance.

We illustrate the proposed formulation by an example. Consider a set of programs P={p1, p2, p3, p4} and a score function S over P such that S(pi) =i. According to our formulation, each program can be abstracted as a probabilistic distribution over P. To specify the distributions, let w_i be a vector of probabilistic weights of length 4 that represents the probabilistic distribution over P corresponding to pi. In this example we set w1= [0.97,0.01,0.01,0.01], w2= [0.75,0,0.25,0], w3= [0.25,0.25,0.25,0.25], w4= [0,0.58,0,0.42].Then a possible RSI procedure may do the flowing. It starts from p3. First p3 generates p4. Since S(p4)> S(p3), the current program is not updated. Then p3 generates p2. The current program is updated to p2 because S(p2)< S(p3). Next p2 generates p1, and the current program updates to p1. Since p1 has the lowest score (highest order), no future program will be updated. Figure 1 shows the corresponding Markov chain.

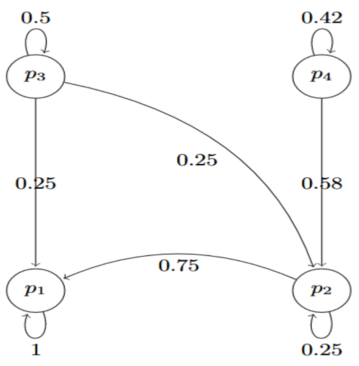


Figure 1 The Markov chain corresponding to the RSI

Fig. 1: The Markov chain corresponding to the RSI procedure defined by given scores and program generation probabilities.

A reasonable utility measure is the expected numbers of steps starting from a program to find the optimal program following our RSI definition. Furthermore, the score function needs to be consistent with the expected numbers of steps from programs to the optimal program following the process defined by itself. We mean that a score function S is consistent if for all p, p′∈P, S(p)> S(p′) implies that the expected number of steps to reach the optimal program from p is greater than starting from p′. More generally, if one takes some measure for a programs’ ability to generate future programs, the score function needs to be consistent with this measure.

Two nice properties hold for this construction. First, the programs are added in a non-decreasing order of scores. Second, the score function equals the expected numbers of steps to reach the optimal program defined by this score function. We will prove the first property. The second property and the consistency of the score function are straightforward from the first property. We describe an example of how such score function is computed given the distributions to generate programs of each program and the optimal program. Consider the same abstraction of programs as the above example, where P={p1, p2, p3, p4} with corresponding probabilistic weights w1= [0.97,0.01,0.01,0.01], w2= [0.75,0,0.25,0], w3= [0.25,0.25,0.25,0.25], w4= [0,0.58,0,0.42]. Fix p1 to be the optimal program. Initially set S(p1) = 0 and S(pi) =∞, i=2,3,4. The transition function of initial Markov chain is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.75 & 0.25 & 0 & 0 \\ 0.25 & 0 & 0.75 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

At the first step, the expected number of steps from p2, p3, p4 following the current Markov chain are 4/3,4,∞. Hence we update S(p2) = 4/3. Because of the change of score, transition of the Markov chain change to

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.75 & 0.25 & 0 & 0 \\ 0.25 & 0.25 & 0.5 & 0 \\ 0 & 0.58 & 0 & 0.42 \end{bmatrix}$$

Then we compute the expected number of steps from p3 and p4 following the updated Markov chain. By some arithmetic we get the expectation are 8/3 for p3 and (approximately) 3.057 for p4. Since 8/3<3.057, update S(p3) = 8/3. By similar procedures, one can compute the score for S(p4).

As shown above, the R-CNN model with different SVM architectures/algorithms to get range of outcomes to get better AI’s with varying degree of accuracy when it is done recursively.

Support Vector Machine (SVM)

The support vector machine (SVM) was developed by for binary classification. Its objective is to find the optimal hyperplane f(w, x)=w·x + b to separate two classes in a given dataset, with features $x \in R^m$.

SVM learns the parameters w by solving an optimization problem (Eq. 1).

$$min \frac{1}{p} w^T w + C \sum_{i=1}^{p} \max(0, 1 - y_i' (w^T x_i + b)) \qquad (1)$$

Where $w^T w$ is the Manhattan norm (also known as L1 norm), C is the penalty parameter (may be an arbitrary value or a selected value using hyper-parameter tuning), y' is the actual label, and $w^T x + b$ is the predictor function. Eq. 1 is known as L1-SVM, with the standard hinge loss. Its differentiable counterpart, L2-SVM (Eq.2), provides more stable results.

$$min \frac{1}{p} \|w\|_2^2 + C \sum_{i=1}^{p} \max(0, 1 - y_i' (w^T x_i + b))^2 \qquad (2)$$

Where ‖w‖2 is the Euclidean norm (also known as L2 norm), with the squared hinge loss.

ARCHITECTURE

Recursive CNN-SVM Architecture For Image Classification

MNIST is an established standard hand written digit classification dataset that is widely used for benchmarking deep learning models. However, we have used the Fashion-MNIST dataset. The said dataset consists of images having the same distribution, the same number of classes, and the same color profile as MNIST. Also, It is having 10,000 training examples and 10,000 test cases.

The Deep Artificial Neural Network

We used two convolutional layers:

- The first layer will have 32-5 x 5 filters,
- The second layer will have 64-5 x 5 filters

In addition, there are two max-pooling layers each of size 2 x 2.

We used a RELU as our activation function which simply takes the output of max_pool and applies RELU.

Fully connected layer:

Just like any other layer, we declare weights and biases as random normal distributions. In fully connected layer, we take all the inputs, do the standard operation on it. The Fully Connected Layer has 1024 Hidden Neurons.

We added Dropout into the network to overcome the problem of overfitting to some extent and also to improve the training and validation accuracy.

The final layer is Output Layer with 10 Output Classes.

At the last layer of the CNN, instead of the conventional softmax function with the cross entropy function (for computing loss), the L2-SVM is implemented. That is, the output shall be translated to the following case $y \in \{-1,+1\}$, and the loss is computed by Eq. 2.The weight parameters are then learned using Adam.

Machine Learning methods for feature selection and classification have been playing active roles in analyzing high-throughput data. We used both normal linear SVM and recursive support vector machine(R-SVM) to select input features for classification. The proposed R-SVM algorithm will recursively classify the training samples with SVM and select features according to their weights in the SVM classifier.

TEST RESULTS

Image Classification is about classifying objects in an image and the test results show good accuracy between training and validation data. However the CNN algorithm needs a lot of regions to predict accurately and hence high computation time.

We compared two methods i.e. SVM and R-SVM , the R-SVM adopting recursive procedures to select features in SVM classifiers. Although the two methods(SVM and R-SVM) did not differ significantly in their validation performances, it appears that R-SVM is more robust and can recover more informative features. The proposed R-SVM method is suitable for analyzing high-throughput data and it outperforms SVM in the robustness and in the ability to recover informative features.

CONCLUSION

Artificial Super Intelligence(ASI) is based on idea that machines not only imitate the human mind but can even supersede human's intelligence. In order to achieve this, ASI will have to be more intelligent by improving intelligent abilities of the artificial machines. In this paper, we have discussed R-CNN model for implementing object detection and implemented CNN-SVM architecture where a lesser AI pass intelligence to high level machine learning algorithm to make better AI recursively. The test results show that it is possible to build artificial machines that have high-level intelligence by combining different intelligent algorithms recursively which are designed to perform specified tasks.

REFERENCES

1. https://github.com/
2. Wenyi Wang " A Formulation of Recursive Self-Improvement and Its Possible Efficiency". https://arxiv.org/pdf/1805.06610.pdf
3. Abien Fred M. Agarap "An Architecture Combining Convolutional Neural Network(CNN) and Support Vector Machine (SVM) for ImageClassification". https://arxiv.org/pdf/1712.03541.pdf