Neural Nets in Embedded Applications

ALVINN: Autonomous Land Vehicle
In a Neural Network
ALVINN Outline

1) How ALVINN Works
   a) Network Architecture
   b) Training "on-the-fly"
   c) Performance

2) Why ALVINN Works
   a) Hidden Unit Analysis
   b) Comparison w/ other techniques

3) Integrating Multiple Networks
   a) Symbolic Arbitration
   b) Connectionist Arbitration

4) Lessons Learned
The architecture is a fully connected 3-layered network. To drive, a the video image is projected onto the input retina. Activation is passed forward through the network and the output unit with the highest activation level determines the direction to steer.
Instead of turning 1 out of 30 output units on,

ALVINN is trained to produce an output vector with a gaussian "hill" of activation centered on the correct steering direction.

This results in better generalization and the ability to make finer steering corrections.
Original Training Scheme

Generate a set of artificial road images resembling the driving situations the network is expected to encounter. Using these images, along with the corresponding correct steering direction for each, as training data. Once the network has reliably learned to produce the correct steering direction on the artificial roads, test it in the real world.

Problem:
Generating realistic artificial road images like the one shown below is a difficult and time consuming process, particularly when many different situations are likely to be encountered.
Train the network to imitate a person’s driving reactions by using live sensor images as input and the person’s current steering direction as the teaching signal.
Training "on the fly": The Problem

The person keeps the vehicle centered on the road and heading in the correct direction during training. As a result, the network never learns to recover after it has made a mistake and strayed from the road center.

Bad Solution:
Have the driver swerve during training to show the network a wider variety of situations. This requires turning learning off to avoid learning incorrect reactions.
Training "on the fly": Good Solution

Increase the training set variety by artificially shifting and rotating the original video image. This makes it appear that the vehicle is at a different orientation relative to the road as shown below.

A simple steering model is used to predict how a person would react to the situations depicted in the transformed images.

![Diagram showing original and transformed images](image-url)
Problem: Forgetting Previous Knowledge

The network still has a tendency to overlearn recently encountered situations and forget how to drive on situations encountered earlier during training.

Example:

After a long right turn, the network will be biased towards turning right, since recent training images were predominantly right turns.
Solution to Forgetting Problem

Maintain buffer of previously encountered road images.

Replace old exemplars in buffer with new ones derived from the current image.

Prevent bias in the buffer by carefully choosing which old exemplars to replace. Can replace either those images which the network has learned the best, or replace old exemplars that have similar steering directions to the new exemplars.

New Exemplars

Buffer

\[ \begin{array}{ccccccc}
\square & \square & \square & \square & \square & \square & \square \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\end{array} \]
Training "on-the-fly": The Details

1) Create 15 shifted and/or rotated training images from the original camera image, altering the correct steering direction appropriately for each.

2) Replace the 15 old exemplars with 15 new exemplars in the 200 element training exemplar buffer.

3) Perform one epoch of back-propagation on the 200 element training exemplar buffer.

4) Repeat 1-3 until network’s dictated steering direction reliably matches person’s steering direction.
ALVINN Performance

Training -- Requires 5 minutes of driving over 300 m of sample road

Testing -- Cycles at up to 20 hz (Digitizing Limited) Drives up to 20 MPH (Navlab limit)

Conditions - Successfully driven on
1) Sunny days
2) Cloudy days
3) Rainy days
4) Snowy days
5) Total darkness

Terrains -- Successfully driven on
1) One lane dirt roads
2) One lane paved roads
3) Multi-lane lined roads
4) Multi-lane unlined roads
5) Parking lot row following
6) Obstacle avoidance
7) Railroad track following
ALVINN is capable of driving in total darkness by using laser reflectance images as input. In a laser reflectence image, the value of a pixel corresponds to how reflective the surface is to infrared laser light. The road and the off-road reflect differently so they can be distinguished WITHOUT AMBIENT LIGHT.
This unit is excited by a road on the left of the image. It suggests a left turn to bring the vehicle back to the center of the road.
This unit is excited by roads slightly left and slightly right of center. It suggests two steering directions, a shallow left and a shallow right turn. In order to determine which is correct, this unit must work together with other units. This kind of distributed representation allows a very simple architecture to drive accurately.
This unit was taken from a network trained on roads whose width varied. In this case, the network found the best representation was for hidden units to detect one edge of the road, and suggest a relatively wide range of steering directions.
ALVINN Weights Evolving

ALVINN weights evolving over time to pick out important image features.

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Hidden Unit 1</th>
<th>Hidden Unit 2</th>
<th>Hidden Unit 3</th>
<th>Hidden Unit 4</th>
<th>Hidden Unit 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison with "Traditional Approach"

In the traditional approach to autonomous navigation, the programmer must:

1) Determine which image features are important (e.g. yellow line down road).

2) Hand-code algorithms to find the important features (e.g. Edge detection to find yellow line).

3) Hand-code algorithm to determine direction to steer from position of features in image.
Comparison with "Traditional Approach"

Artificial Neural Network advantages over traditional approach to autonomous navigation:

1) Simple, regular processing results in fast cycle time, and higher driving speeds. ALVINN drives NAVLAB between 3 and 8 times faster than hand-coded algorithms.

2) ALVINN learns which features are important, make it more flexible than hand programmed algorithms. As a result, ALVINN has driven in more situations than any other autonomous navigation system.

3) ALVINN doesn't rely on finding a single image feature, but combines all relevant features to determine steering direction, making it more robust than hand-coded algorithms.
Comparison with Nearest Neighbor Matching

Matches on spurious features, like a car in the periphery of the image, can result in steering errors. ALVINN learns key on only those features which are important, avoiding this kind of mistake.

Current Image

![Current Image Diagram]

Templates

![Template Diagrams]

Best Match
ALVINVN Shortcomings

The single network ALVINVN architecture can only drive on a single type of road, and can't transition from one road type to another.

ALVINVN can't plan and follow a route. The a driving network will keep the vehicle on the road, but has trouble making decisions at intersections.

Solution:
Rule-based multi-network integration
Multiple networks provide steering directions to the arbitrator, which decides which one to listen to and therefore where to steer. The mapper helps the arbitrator make its decision by providing it with symbolic information about the path to be followed.
Symbolic Mapping Module

The symbolic mapping module maintains the vehicle's position on a map and provides two pieces of information to the arbitrator. It provides a steering direction to follow the pre-planned path, and it provides data on the current terrain.

![Diagram of a map showing current and ending points, with different road types and landmarks indicated.]
Relevancy Arbitration

One technique the symbolic arbitrator uses to decide which network to listen to is called relevancy arbitration. In this technique, the arbitrator chooses the knowledge source to listen to based on the terrain data provided by the mapping module.

Examples:

If the mapping module says the vehicle is currently on a two lane road, the arbitrator will choose to listen to the two lane road driving network.

If the mapping module says the vehicle is coming up to an intersection, the arbitrator will choose to steer in the direction dictated by the mapping module in order to follow the path to the destination.
Priority Arbitration

Another rule-based arbitration technique used in the hybrid ALVINN system is called priority arbitration. In this technique, the importance or urgency of the networks' outputs are ranked. The arbitrator chooses to steer in the direction dictated by the network with the most urgent output.

Example:

The obstacle avoidance network is trained to steer straight ahead unless there is an obstacle directly ahead of the vehicle, in which case it is trained to swerve to avoid the obstacle.

Using priority arbitration, the arbitrator will ignore the obstacle avoidance network when it says steer straight ahead, since the message has low urgency.

But when the obstacle avoidance network says turn sharply, its urgency is high, since obstacle avoidance take precedence over other behaviors, so the arbitrator chooses to steer as the obstacle avoidance network indicates.
Obstacle Avoidance Network

The obstacle avoidance network receives input from the scanning laser rangefinder, which shows distance to objects in the scene. It is trained to steer straight ahead when the path ahead is clear, and to swerve to avoid any obstacles in the path of the vehicle.
Problems with Rule-Based Arbitration

Relevancy and priority arbitration choose a single network to listen to. They do not combine the outputs from multiple networks. This is particularly bad when multiple networks are trained to drive in the same situation using different sensor modalities.
Other Problems with Rule-Based Arbitration

Requires detailed knowledge of networks' areas of expertise.

Requires a very detailed and accurate map of the environment. Also requires precise knowledge of the vehicle's current position.

We would like a system which could follow vague directions like:

"Go about 1/4 mile and turn right at the intersection."
Connectionist Arbitration

Idea:

Estimate the reliability of all networks on the current situation.

Use the reliability estimate to:
1) weight the outputs of the networks
2) Pinpoint the vehicle’s map location
3) Control the vehicle’s speed
4) Determine need for retraining

But how do you estimate a network’s reliability?
Output Appearance Reliability Estimation

Idea:

Compare actual output with nearest "ideal" output. The bigger this difference, called the "appearance error" the less reliable the network.

Actual Output Response

Nearest Ideal Output Response
Output Appearance Reliability Estimation

\[ \text{Appearance Error} = \sum_{i} (\text{Actual}_i - \text{Ideal}_i)^2 \]

Steering Error = \left| \text{curve}_h - \text{curve}_n \right|

where \( \text{curve}_h \) = human turn curvature
and \( \text{curve}_n \) = network turn curvature

Correlation Coefficient = 0.78

Intersection of steering error and output appearance error
When ALVINN reaches a fork in the road, the two branches in the image result in a bimodal output vector. The high appearance error indicates the vehicle has reached the intersection. The vague directions (e.g. turn right at the intersection) are then used to bias which of the two steering directions is chosen.
Comparative Output Appearance Error

As the road changes from one type to another, the appearance errors of networks changes dramatically. Such changes can be used to pinpoint the vehicle’s location.

Since there is such a large difference between the appearance errors of the networks, it is easy to choose which net (or nets) to listen to.

For smoothly combining the outputs from multiple networks, the appearance error is used as a weighting function.
Speed Control

If none of the networks appear to be reliable in the current situation, the system slows down the vehicle and may even ask for additional training on the new situation.

Other ways ALVNN controls speed:

If networks are steering erratically, slow down.

If networks are steering sharply, slow down.
Lessons Learned

Preprocessing Tradeoff - The more preprocessing done, the more likely the net will solve difficult problems, but the less flexible the net will be when faced with new situations.

Importance of Modularity - By requiring individual networks to handle relatively restricted situations, network training becomes faster and more robust.

Viability of Hybrid Approach - To achieve high level behavior like route following, neural networks can (and currently must) be combined with symbolic techniques.

Not Just Black Boxes - Neural networks can be analyzed to determine how they are working (more later), and the answers they produced can be analyzed to determine their reliability.
ALVINN Bibliography


Neural Nets in Embedded Applications

Neural Network Vision Guidance for the Self Mobile Space Manipulator
The Robot and Its Job

The Self Mobile Space Manipulator (SM^2) is a two-legged robot designed to walk on the exterior of the space station. Its designed to transport parts, perform visual inspection, and light construction.
Flexibility Problem

The robot's large size and lightweight construction make it difficult to precisely position the tip (or foot) of the robot for anchoring purposes.
Solution to Flexibility Problem

Put video cameras on the robot’s foot, and use an artificial neural network to determine the foot’s position relative to the anchor hole.
Network Architecture

The network is a fully connected 3-layered network. The output represents the foot displacement relative to the anchor hole in the X and Y directions.
Results

1) Training takes 10 min. and ~300 images.

2) Cycles at 15 frames/sec. once trained.

3) Results in ~5 times fewer missed insertions than without vision feedback.

4) Allows robot to perform robust 3D walking
What the Network Sees

This figure illustrates the low resolution video image the network receives as input, and the network's response. In this image, the tip is centered over the anchor hole, so the network responds with 0.0 offset in both the X and Y dimensions.
Comparison with Traditional Approach

A) Traditional vision approach
   1) Put target next to anchor hole
      a) white-on-black cross
      b) red ring
   2) Use image processing techniques such as thresholding and simple pattern matching to find target

B) Comparison
   1) NNs easier to develop and retrain
   2) NNs don’t require altering space station
   3) NNs can handle other tasks like grabbing door handle, tracking strut.
Network Weights Evolving

This figure shows the weights from the input retina to each of the five hidden units at four points during training. By epoch 100, the network has developed detectors with both vertical and horizontal orientations.
Conclusion

The techniques developed in ALVINN for robot driving are also applicable to other forms of vision-based robot guidance.

The simplicity of adapting ALVINN to this new domain underscores the flexibility advantage neural networks have over hand-coded systems.