**Ιστορικό τεχνητών νευρικών δικτύων**

Ένα τεχνητό νευρικό δίκτυο είναι ένα μαθηματικό μοντέλο υπολογιστών που έχει σχεδιαστεί για να μιμείται τον τρόπο με τον οποίο ο εγκέφαλος αντιδρά στις αισθητηριακές εισόδους. Ο εγκέφαλος αποτελείται από εκατομμύρια νευρώνες που συνδέονται μεταξύ τους σε τεράστια δίκτυα. Κάθε νευρώνας μπορεί να διεγείρεται, σαν να ενεργοποιείται ή απενεργοποιείται ένας διακόπτης και η κατάσταση του νευρώνα ενεργοποιεί ή απενεργοποιεί επίσης τους γύρω νευρώνες ανάλογα με το επίπεδο ενεργοποίησης του νευρώνα και την ισχύ της σύνδεσης μεταξύ των νευρώνων . Ένας νευρώνας με ισχυρή σύνδεση θα έχει μεγαλύτερο επίπεδο διέγερσης από έναν με ασθενέστερη σύνδεση. Πολύ απλοϊκά, το επίπεδο διέγερσης των νευρώνων σχετίζεται με το άθροισμα της διέγερσης που λαμβάνει από όλους τους άλλους νευρώνες που συνδέονται με αυτό, και έτσι ακριβώς λειτουργεί το τεχνητό νευρικό δίκτυο.

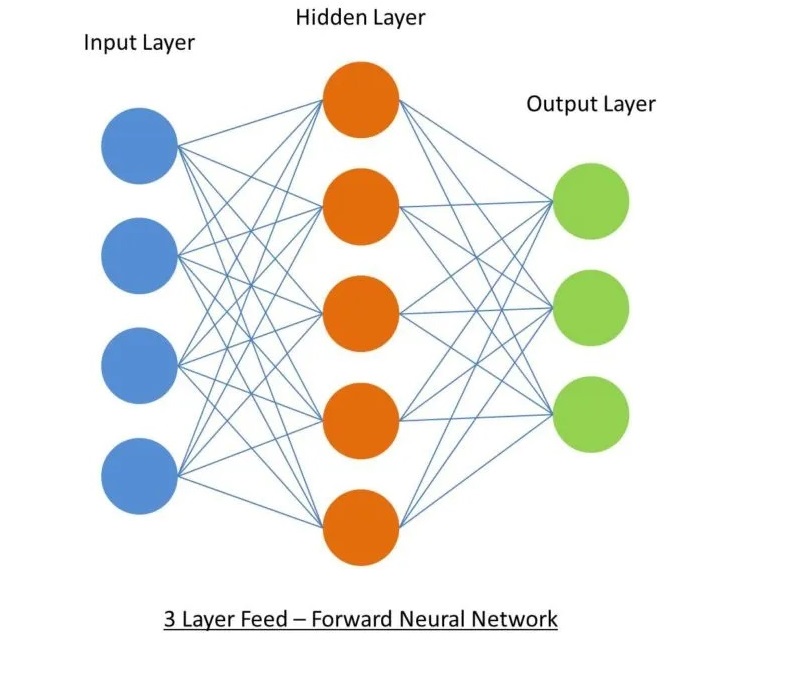
Ας ξεκινήσουμε καταλαβαίνοντας τι ακριβώς είναι ένα δίκτυο backpropagation. Το ίδιο το δίκτυο δεν είναι μια νέα αντίληψη, στην πραγματικότητα βρίσκονται από τη δεκαετία του '80 και ενώ βασίζονται σε αρκετά περίπλοκα μαθηματικά, δεν χρειάζεται να καταλάβετε τα μαθηματικά για να καταλάβετε πώς λειτουργεί το δίκτυο.

Τι είναι λοιπόν ένα τεχνητό νευρικό δίκτυο; Εν ολίγοις, ένα τεχνητό νευρωνικό δίκτυο είναι ένα τμήμα κώδικα που μαθαίνει πώς να ανταποκρίνεται σε εισόδους με βάση παραδείγματα συνόλων εισόδων και εξόδων. Είναι πολύ ισχυρά εργαλεία και βρίσκουν γρήγορα τη θέση τους στην αναγνώριση προσώπου, τα αυτόνομα οχήματα, τις χρηματιστηριακές προβλέψεις και τις αθλητικές προβλέψεις, ακόμα και σε ιστότοπους που προτείνουν προϊόντα που μπορεί να σας ενδιαφέρουν. Η πιο ισχυρή εφαρμογή τους έγκειται στην αναγνώριση προτύπων, όπου η ακριβής η είσοδος στο δίκτυο δεν είναι γνωστή. Μπορεί να υπάρχουν πολύ λίγες ή πάρα πολλές πληροφορίες και εναπόκειται στο δίκτυο να αποφασίσει πώς θα υποβληθεί σε επεξεργασία. Ένα καλό παράδειγμα εφαρμογής ενός τεχνητού νευρικού δικτύου είναι η αναγνώριση χειρογράφου. Είμαστε σε θέση να αναγνωρίσουμε γράμματα και αριθμούς, αλλά το ακριβές σχήμα των χαρακτήρων ποικίλλει από άτομο σε άτομο, Επομένως, η είσοδος στο νευρικό δίκτυο δεν είναι ποτέ γνωστή με ακρίβεια. Εναπόκειται στο νευρωνικό δίκτυο να προσδιορίσει την είσοδο και να τη συσχετίσει με τη σχετική έξοδο.

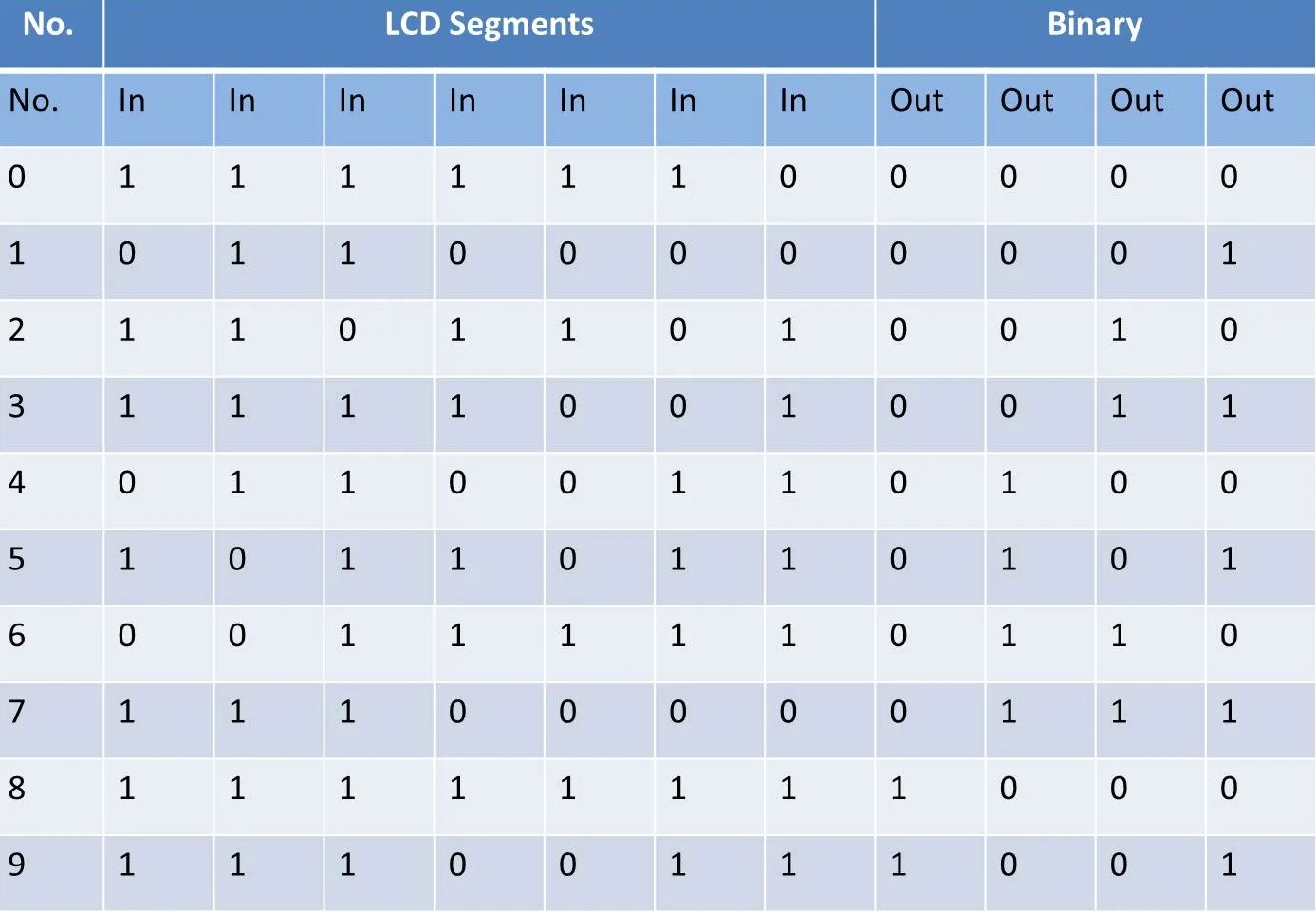
Σε ένα τεχνητό ή βασισμένο σε λογισμικό νευρωνικό δίκτυο, δημιουργείται ένα μαθηματικό μοντέλο όλων των νευρώνων και των συνδέσεών τους. Στη συνέχεια μια είσοδος τροφοδοτείται στο δίκτυο και οι νευρώνες προσθέτουν συστηματικά τις εισόδους τους και παράγουν μια έξοδο στο επόμενο επίπεδο των νευρώνων έως ότου επιτευχθεί μια έξοδος.

Μία από τις βασικές αρχές ενός τεχνητού νευρικού δικτύου είναι ότι το δίκτυο πρέπει να εκπαιδευτεί. Όταν το δίκτυο έχει ρυθμιστεί, εφαρμόζονται τυχαία βάρη σε κάθε μία από τις συνδέσεις. Αυτά τα βάρη παρέχουν ένα σημείο εκκίνησης για το δίκτυο, αλλά σχεδόν πάντα θα παρέχουν εξόδους "σκουπίδια". Ένα σύνολο δειγμάτων δεδομένων εισάγεται στο δίκτυο και τα αποτελέσματα συγκρίνονται με τα αναμενόμενα αποτελέσματα. Τα βάρη στη συνέχεια ρυθμίζονται και ο κύκλος εισόδου / εξόδου επαναλαμβάνεται. Αυτός ο κύκλος εκπαίδευσης επαναλαμβάνεται έως ότου τα δεδομένα εξόδου από το δίκτυο ταιριάζουν με τα αναμενόμενα δεδομένα εξόδου εντός ενός συγκεκριμένου επιπέδου ακρίβειας. Αυτό διαρκεί συνήθως μερικές δεκάδες χιλιάδες κύκλους εκπαίδευσης ανάλογα με την πολυπλοκότητα των δεδομένων και του δικτύου.

Σε αυτό το παράδειγμα, πρόκειται να οικοδομήσουμε ένα δίκτυο τροφοδοσίας τριών επιπέδων, με τα τρία να είναι το επίπεδο εισόδου, το κρυφό επίπεδο και το επίπεδο εξόδου όπως φαίνεται παρακάτω.



Στο σχέδιο σε αυτό το άρθρο παρουσιάζεται ένα σύνολο εκπαιδευτικών εισόδων και εξόδων που αντιστοιχούν τα επτά τμήματα μιας αριθμητικής οθόνης LED στον αντίστοιχο δυαδικό αριθμό. Το δίκτυο διατρέχει τα δεδομένα εκπαίδευσης επαναλαμβανόμενα και πραγματοποιεί προσαρμογές στους συντελεστές στάθμισης έως ότου επιτευχθεί ένα συγκεκριμένο επίπεδο ακρίβειας, σε αυτό το στάδιο το δίκτυο λέγεται ότι έχει εκπαιδευτεί.Τα δεδομένα εκπαίδευσης εισόδου:

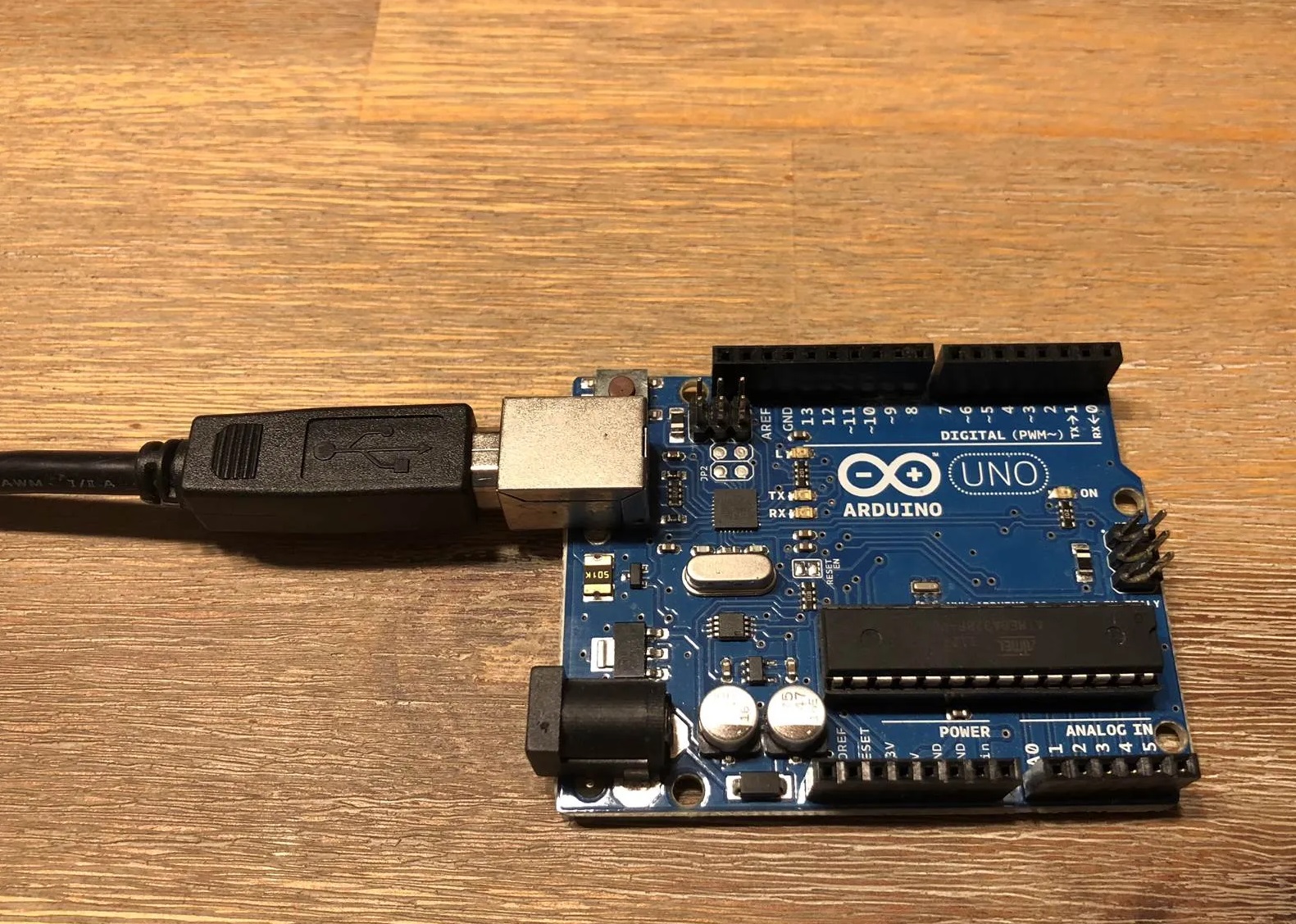


Θα χρειαστεί να εκτελέσετε το Serial monitor στο Arduino IDE για να δείτε την προοδευτική εκπαίδευση και τα τελικά αποτελέσματα. Το πρόγραμμα θα στέλνει μέσω ενός συνόλου δεδομένων εκπαίδευσης κάθε χίλιους κύκλους, ώστε να μπορείτε να δείτε πώς το δίκτυο «μαθαίνει» και πλησιάζει στις σωστές απαντήσεις.

Μπορείτε να δημιουργήσετε τα δικά σας εκπαιδευτικά δεδομένα για να εκπαιδεύσετε το δίκτυό σας, ρίξτε μια ματιά στις τελευταίες ενότητες σε αυτόν τον οδηγό για οδηγίες σχετικά με τη δημιουργία των δικών σας εκπαιδευτικών δεδομένων.

**Τι χρειάζεστε για το τεχνητό νευρωνικό δίκτυο που βασίζεται στο Arduino**

* Ένα Arduino (Uno χρησιμοποιείται σε αυτόν τον οδηγό)

**Πώς να φτιάξετε το τεχνητό νευρικό σας δίκτυο με βάση το Arduino**

Σε αυτό το παράδειγμα, απλώς εκπαιδεύουμε ένα δίκτυο με ένα προκαθορισμένο σύνολο δεδομένων εκπαίδευσης έως ότου επιτευχθεί μια λύση. Αυτός είναι ο ευκολότερος και πιο βασικός τρόπος για να εκτελέσετε ένα τεχνητό νευρικό δίκτυο στο Arduino σας και δεν απαιτεί συνδέσεις με τους ακροδέκτες εισόδου ή εξόδου. Απλά θα πρέπει να συνδέσετε το Arduino στον υπολογιστή σας χρησιμοποιώντας το καλώδιο USB και είστε έτοιμοι να ανεβάσετε τον κωδικό νευρωνικού δικτύου.

**Κατανόηση του κώδικα**

Όπως αναφέρθηκε προηγουμένως, τα μαθηματικά πίσω από ένα νευρωνικό δίκτυο μπορεί να είναι αρκετά περίπλοκα εάν δεν έχετε ισχυρό μαθηματικό υπόβαθρο, αλλά ευτυχώς δεν χρειάζεται να καταλάβετε τον κώδικα για να μπορείτε να τον χρησιμοποιήσετε και να τον τροποποιήσετε για να χρησιμοποιήσετε τα δικά σας εκπαιδευτικά δεδομένα . Θα πρέπει να μπορείτε να ακολουθείτε την πλειοψηφία του κώδικα μέσω μιας απλής κατανόησης των συστοιχιών και των βρόχων.

Απλά, το πρόγραμμα δημιουργεί ένα σύστημα συστοιχιών που αποθηκεύουν τα βάρη του δικτύου και τα δεδομένα που τροφοδοτούνται μέσω του δικτύου. Τα δεδομένα στη συνέχεια τροφοδοτούνται διαδοχικά μέσω του δικτύου και στη συνέχεια τα σφάλματα μεταδίδονται ξανά μέσω του δικτύου και προσαρμόζονται οι σταθμίσεις.

Ακολουθεί μια σύνοψη της λειτουργίας κώδικα:

* Ρυθμίστε τις συστοιχίες και αντιστοιχίστε τυχαία βάρη.
* Ξεκινήστε έναν βρόχο που διατρέχει κάθε στοιχείο των εκπαιδευτικών δεδομένων.
* Τυχαιοποιήστε τη σειρά με την οποία εκτελούνται τα εκπαιδευτικά δεδομένα σε κάθε επανάληψη για να διασφαλιστεί ότι δεν θα πραγματοποιηθεί σύγκλιση στα τοπικά ελάχιστα.
* Τροφοδοτήστε τα δεδομένα μέσω του δικτύου υπολογίζοντας την ενεργοποίηση των κόμβων του κρυφού επιπέδου, τους κόμβους του επιπέδου εξόδου και τα σφάλματα.
* Πίσω διαδώστε τα σφάλματα στο κρυφό επίπεδο.
* Ενημερώστε τα σχετικά βάρη.
* Συγκρίνετε το σφάλμα με το όριο και αποφασίστε εάν θα εκτελέσετε έναν άλλο κύκλο ή εάν η προπόνηση έχει ολοκληρωθεί.
* Στείλτε ένα δείγμα των δεδομένων εκπαίδευσης στη Σειριακή οθόνη κάθε χίλιους κύκλους.

**Μεταφόρτωση του κώδικα**

Ο καλύτερος τρόπος για να μάθετε και να καταλάβετε πώς λειτουργεί ο κώδικας είναι να τον εκτελέσετε και να δείτε στη Σειριακή οθόνη πώς αναπτύσσεται η λύση στα εκπαιδευτικά δεδομένα.

Εδώ είναι ο κωδικός:

// 1

# συμπερίληψη <math.h>

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Διαμόρφωση δικτύου - προσαρμοσμένο ανά δίκτυο

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

const int PatternCount = 10;

const int InputNodes = 7;

const int HiddenNodes = 8;

const int OutputNodes = 4;

const float LearningRate = 0,3;

const float Momentum = 0,9;

const float InitialWeightMax = 0,5;

const float Επιτυχία = 0,0004;

const byte Είσοδος [PatternCount] [InputNodes] = {

{1, 1, 1, 1, 1, 1, 0}, // 0

{0, 1, 1, 0, 0, 0, 0}, // 1

{1, 1, 0, 1, 1, 0, 1}, // 2

{1, 1, 1, 1, 0, 0, 1}, // 3

{0, 1, 1, 0, 0, 1, 1}, // 4

{1, 0, 1, 1, 0, 1, 1}, // 5

{0, 0, 1, 1, 1, 1, 1}, // 6

{1, 1, 1, 0, 0, 0, 0}, // 7

{1, 1, 1, 1, 1, 1, 1}, // 8

{1, 1, 1, 0, 0, 1, 1} // 9

};

const byte Στόχος [PatternCount] [OutputNodes] = {

{0, 0, 0, 0},

{0, 0, 0, 1},

{0, 0, 1, 0},

{0, 0, 1, 1},

{0, 1, 0, 0},

{0, 1, 0, 1},

{0, 1, 1, 0},

{0, 1, 1, 1},

{1, 0, 0, 0},

{1, 0, 0, 1}

};

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Τερματισμός διαμόρφωσης δικτύου

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

int i, j, p, q, r;

int ReportEvery1000;

int RandomizedIndex [PatternCount];

μακρύς κύκλος προπόνησης;

επιπλέουν Rando?

Σφάλμα float;

Float Accum;

float Hidden [HiddenNodes];

Float Output [OutputNodes];

float HiddenWeights [InputNodes + 1] [HiddenNodes];

float OutputWeights [HiddenNodes + 1] [OutputNodes];

float HiddenDelta [HiddenNodes];

float OutputDelta [OutputNodes];

float ChangeHiddenWeights [InputNodes + 1] [HiddenNodes];

float ChangeOutputWeights [HiddenNodes + 1] [OutputNodes];

άκυρη ρύθμιση () {

Serial.begin (9600);

randomSeed (analogRead (3));

ReportEvery1000 = 1;

για (p = 0; p <PatternCount; p ++) {

RandomizedIndex [p] = p;

}

}

κενός βρόχος () {

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Αρχικοποιήστε HiddenWeights και ChangeHiddenWeights

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (i = 0; i <HiddenNodes; i ++) {

για (j = 0; j <= InputNodes; j ++) {

ChangeHiddenWeights [j] [i] = 0,0;

Rando = float (τυχαίο (100)) / 100;

HiddenWeights [j] [i] = 2.0 \* (Rando - 0.5) \* InitialWeightMax;

}

}

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Αρχικοποιήστε το OutputWeights και το ChangeOutputWeights

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (i = 0; i <OutputNodes; i ++) {

για (j = 0; j <= HiddenNodes; j ++) {

ChangeOutputWeights [j] [i] = 0,0;

Rando = float (τυχαία (100)) / 100;

OutputWeights [j] [i] = 2,0 \* (Rando - 0,5) \* InitialWeightMax;

}

}

Serial.println ("Αρχικές / μη εκπαιδευμένες έξοδοι:");

toTerminal ();

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Ξεκινήστε την προπόνηση

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (TrainingCycle = 1; TrainingCycle <2147483647; TrainingCycle ++) {

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Τυχαιοποιήστε τη σειρά των προτύπων εκπαίδευσης

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (p = 0; p <PatternCount; p ++) {

q = τυχαία (PatternCount);

r = RandomizedIndex [p];

RandomizedIndex [p] = RandomizedIndex [q];

RandomizedIndex [q] = r;

}

Σφάλμα = 0,0;

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Κάντε κύκλο σε κάθε σχέδιο προπόνησης με τυχαία σειρά

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (q = 0; q <PatternCount; q ++) {

p = RandomizedIndex [q];

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Υπολογίστε κρυφές στρώσεις ενεργοποιήσεις

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (i = 0; i <HiddenNodes; i ++) {

Accum = HiddenWeights [InputNodes] [i];

για (j = 0; j <InputNodes; j ++) {

Accum + = Είσοδος [p] [j] \* HiddenWeights [j] [i];

}

Κρυφό [i] = 1.0 / (1.0 + exp (-Accum));

}

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Υπολογίστε τις ενεργοποιήσεις του επιπέδου εξόδου και υπολογίστε τα λάθη

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (i = 0; i <OutputNodes; i ++) {

Accum = OutputWeights [HiddenNodes] [i];

για (j = 0; j <HiddenNodes; j ++) {

Accum + = Κρυφό [j] \* OutputWeights [j] [i];

}

Έξοδος [i] = 1.0 / (1.0 + exp (-Accum));

OutputDelta [i] = (Στόχος [p] [i] - Έξοδος [i]) \* Έξοδος [i] \* (1.0 - Έξοδος [i]);

Σφάλμα + = 0,5 \* (Στόχος [p] [i] - Έξοδος [i]) \* (Στόχος [p] [i] - Έξοδος [i]);

}

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Σφάλματα Backpropagate σε κρυφό επίπεδο

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (i = 0; i <HiddenNodes; i ++) {

Συγκέντρωση = 0,0;

για (j = 0; j <OutputNodes; j ++) {

Accum + = OutputWeights [i] [j] \* OutputDelta [j];

}

HiddenDelta [i] = Accum \* Hidden [i] \* (1.0 - Hidden [i]);

}

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Ενημέρωση εσωτερικού -> Κρυμμένα βάρη

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (i = 0; i <HiddenNodes; i ++) {

ChangeHiddenWeights [InputNodes] [i] = LearningRate \* HiddenDelta [i] + Momentum \* ChangeHiddenWeights [InputNodes] [i];

HiddenWeights [InputNodes] [i] + = ChangeHiddenWeights [InputNodes] [i];

για (j = 0; j <InputNodes; j ++) {

ChangeHiddenWeights [j] [i] = LearningRate \* Input [p] [j] \* HiddenDelta [i] + Momentum \* ChangeHiddenWeights [j] [i];

HiddenWeights [j] [i] + = ChangeHiddenWeights [j] [i];

}

}

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Ενημέρωση κρυφό -> Βάρη εξόδου

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (i = 0; i <OutputNodes; i ++) {

ChangeOutputWeights [HiddenNodes] [i] = LearningRate \* OutputDelta [i] + Momentum \* ChangeOutputWeights [HiddenNodes] [i];

OutputWeights [HiddenNodes] [i] + = ChangeOutputWeights [HiddenNodes] [i];

για (j = 0; j <HiddenNodes; j ++) {

ChangeOutputWeights [j] [i] = LearningRate \* Hidden [j] \* OutputDelta [i] + Momentum \* ChangeOutputWeights [j] [i];

OutputWeights [j] [i] + = ChangeOutputWeights [j] [i];

}

}

}

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Κάθε 1000 κύκλοι στέλνουν δεδομένα στο τερματικό για προβολή

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

ReportEvery1000 = ReportEvery1000 - 1;

εάν (ReportEvery1000 == 0)

{

Serial.println ();

Serial.println ();

Serial.print ("TrainingCycle:");

Serial.print (TrainingCycle);

Serial.print ("Σφάλμα =");

Serial.println (Σφάλμα, 5);

toTerminal ();

εάν (TrainingCycle == 1)

{

ReportEvery1000 = 999;

}

αλλού

{

ReportEvery1000 = 1000; Αναφορά

}

}

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Εάν το ποσοστό σφάλματος είναι μικρότερο από το προκαθορισμένο όριο, τότε τερματίστε

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

εάν (Σφάλμα <Επιτυχία) διακοπή;

}

Serial.println ();

Serial.println ();

Serial.print ("TrainingCycle:");

Serial.print (TrainingCycle);

Serial.print ("Σφάλμα =");

Serial.println (Σφάλμα, 5);

toTerminal ();

Serial.println ();

Serial.println ();

Serial.println ("Σετ εκπαίδευσης επιλυμένο!");

Serial.println ("--------");

Serial.println ();

Serial.println ();

ReportEvery1000 = 1;

}

άκυρο προς Τερματικό ()

{

για (p = 0; p <PatternCount; p ++) {

Serial.println ();

Serial.print ("Σχέδιο προπόνησης:");

Serial.println (σελ);

Serial.print ("Εισαγωγή");

για (i = 0; i <InputNodes; i ++) {

Serial.print (Είσοδος [p] [i], DEC);

Serial.print ("");

}

Serial.print ("Στόχος");

για (i = 0; i <OutputNodes; i ++) {

Serial.print (Στόχος [p] [i], DEC);

Serial.print ("");

}

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Υπολογίστε κρυφές στρώσεις ενεργοποιήσεις

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (i = 0; i <HiddenNodes; i ++) {

Accum = HiddenWeights [InputNodes] [i];

για (j = 0; j <InputNodes; j ++) {

Accum + = Είσοδος [p] [j] \* HiddenWeights [j] [i];

}

Κρυφό [i] = 1.0 / (1.0 + exp (-Accum));

}

/ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Υπολογίστε τις ενεργοποιήσεις του επιπέδου εξόδου και υπολογίστε τα λάθη

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /

για (i = 0; i <OutputNodes; i ++) {

Accum = OutputWeights [HiddenNodes] [i];

για (j = 0; j <HiddenNodes; j ++) {

Accum + = Κρυφό [j] \* OutputWeights [j] [i];

}

Έξοδος [i] = 1.0 / (1.0 + exp (-Accum));

}

Serial.print ("Έξοδος");

για (i = 0; i <OutputNodes; i ++) {

Serial.print (Έξοδος [i], 5);

Serial.print ("");

}

}

}

**Δημιουργία των δικών σας εκπαιδευτικών δεδομένων**

Μόλις εκτελέσετε το βασικό δίκτυο, ίσως θελήσετε να δοκιμάσετε και να εισαγάγετε τα δικά σας εκπαιδευτικά δεδομένα. Για να το κάνετε αυτό, θα πρέπει να τροποποιήσετε τα δεδομένα εκπαίδευσης στον πίνακα καθώς και αυτά τα στοιχεία στις παραμέτρους εισαγωγής:

* PatternCount - Ο αριθμός των στοιχείων / σειρά των εκπαιδευτικών δεδομένων στον πίνακα σας.
* InputNodes - Ο αριθμός των νευρώνων που σχετίζονται με τα δεδομένα εισόδου.
* Κόμβοι εξόδου - Ο αριθμός των νευρώνων που σχετίζονται με τα δεδομένα εξόδου.

Εκτός από τις παραπάνω παραμέτρους που πρέπει να αλλάξουν για τα νέα δεδομένα εκπαίδευσης, τα ακόλουθα στοιχεία μπορούν επίσης να αλλάξουν και να πειραματιστούν για να λάβουν διαφορετικά αποτελέσματα εκπαίδευσης:

* HiddenNodes - Ο αριθμός των νευρώνων που σχετίζονται με το κρυφό στρώμα.
* LearningRate - Το ποσοστό του σφάλματος που διαδίδεται ξανά.
* Momentum - Το ποσοστό της προηγούμενης επανάληψης που επηρεάζει την τρέχουσα επανάληψη.
* InitialWeightMax - Η μέγιστη τιμή εκκίνησης για τα τυχαία αντιστοιχισμένα βάρη.
* Επιτυχία - Το κατώφλι στο οποίο το πρόγραμμα αναγνωρίζει ότι έχει εκπαιδευτεί επαρκώς.

Ίσως χρειαστεί να προσαρμόσετε μερικές ή όλες αυτές τις τιμές για να βελτιστοποιήσετε τη διαδικασία εκπαίδευσης για τα νέα σας δεδομένα εκπαίδευσης. Είναι πιθανό μια λύση να μην επιτευχθεί ποτέ και η διαδικασία προπόνησης να κολλήσει ταλαντευόμενη πάνω και κάτω από το κατώτατο όριο, θα πρέπει στη συνέχεια να προσαρμόσετε αυτές τις τιμές έτσι ώστε να είναι δυνατή η επίτευξη μιας λύσης. Μπορείτε να διαβάσετε περαιτέρω για καθεμία από αυτές τις παραμέτρους εάν ερευνήσετε και βελτιώσετε την κατανόησή σας σχετικά με τον τρόπο λειτουργίας των τεχνητών νευρικών δικτύων.

Αξίζει να σημειωθεί ότι τα δεδομένα εκπαίδευσης και η διαμόρφωση δικτύου που παρέχονται σε αυτό το παράδειγμα είναι περίπου τόσο μεγάλες όσο μπορείτε να εκτελέσετε σε ένα Arduino Uno χωρίς να υπερβείτε τα 2K SRAM. Εάν θέλετε να πειραματιστείτε με ένα μεγαλύτερο δίκτυο, θα πρέπει να χρησιμοποιήσετε έναν πίνακα Arduino με μεγαλύτερη κατανομή SRAM, όπως το Mega. Δυστυχώς, δεν δίνεται προειδοποίηση από το IDE ή το Arduino εάν ξεπεραστεί η κατανομή, θα συνεχίσετε να παίρνετε παράξενα αποτελέσματα και το δίκτυο δεν θα είναι σε θέση να εκπαιδευτεί.

**Εμπόδιο που αποφεύγει το ρομπότ να τρέχει ένα νευρικό δίκτυο**

Ο Tim Kälin έχει χρησιμοποιήσει αυτόν τον κωδικό ως βάση για ένα ρομπότ αποφυγής εμποδίων που χρησιμοποιεί δύο υπερηχητικές μονάδες συνδεδεμένες σε ένα ESP32 που εκτελεί το νευρικό δίκτυο για να ελέγχει τις κινήσεις του. Το νευρωνικό δίκτυο έχει δύο κόμβους εισόδου από τις υπερηχητικές μονάδες και πέντε κόμβους εξόδου, στρίψτε αριστερά, στρίψτε δεξιά, ελαφριά αριστερά, ελαφριά δεξιά και πηγαίνετε ευθεία

Σύμφωνα με τον Tim, χρειάζονται ένα ή δύο λεπτά για να εκπαιδεύσει το νευρωνικό δίκτυο στο σύνολο δεδομένων κατά την ενεργοποίηση. Διαθέτει επίσης μερικές ενσωματωμένες λωρίδες LED που χρησιμοποιούνται για την εμφάνιση των εισόδων, της ενεργοποίησης κρυφών επιπέδων και επιλεγμένων εξόδων, έτσι ώστε να έχετε οπτική αναπαράσταση της λειτουργίας του νευρικού δικτύου.

Είναι πραγματικά εντυπωσιακό να παρακολουθείτε.

Ο κώδικας είναι λίγο πιο περίπλοκος από αυτό σε αυτό το παράδειγμα λόγω της πολυπλοκότητας του έργου, αλλά είναι κατανεμημένος αρκετά καλά. Μπορείτε να κατεβάσετε τον κωδικό μέσω αυτού του συνδέσμου - [Neural Network Robot](https://www.the-diy-life.com/wp-content/uploads/2018/06/Neural-Network-Robot.zip)

**Τι να δοκιμάσετε στη συνέχεια**

Μόλις εξοικειωθείτε με τα τεχνητά νευρικά δίκτυα και δοκιμάσετε να πειραματιστείτε με διαφορετικά δεδομένα εκπαίδευσης, είμαι βέβαιος ότι θα θέλατε να κάνετε χρήση του Arduino με πιο πρακτικό τρόπο. Ακολουθούν μερικές ιδέες για να προωθήσουμε αυτό το έργο:

* [Προσθέστε μια οθόνη LCD](https://www.the-diy-life.com/connect-an-lcd-screen-to-an-arduino/) ή TFT στο Arduino σας και στείλτε τα δεδομένα εκπαίδευσης ή εξόδου στην οθόνη αντί για τη σειριακή οθόνη σας.
* Αναπτύξτε ένα δίκτυο που ανταποκρίνεται σε εισόδους στο Arduino. Για παράδειγμα, θα μπορούσατε να χρησιμοποιήσετε φυσικούς διακόπτες ή φωτοαντιστάκτες στις εισόδους Arduino για να ενεργοποιήσετε τους κόμβους εισόδου και να οδηγήσετε μια έξοδο "μάθησης".
* Χρησιμοποιήστε το δίκτυο για να αυξήσετε τις εξόδους στο Arduino σας. Προσθέστε έναν κινητήρα ή σερβο στο Arduino που χρησιμοποιεί το νευρωνικό δίκτυο για να ανταποκριθεί στις εισόδους. Για παράδειγμα, θα μπορούσατε να δημιουργήσετε μια οθόνη σκιάς σερβο βραχίονα που καλύπτει το Arduino όταν το φως πέφτει πάνω σε ένα φωτοαντίσταση.