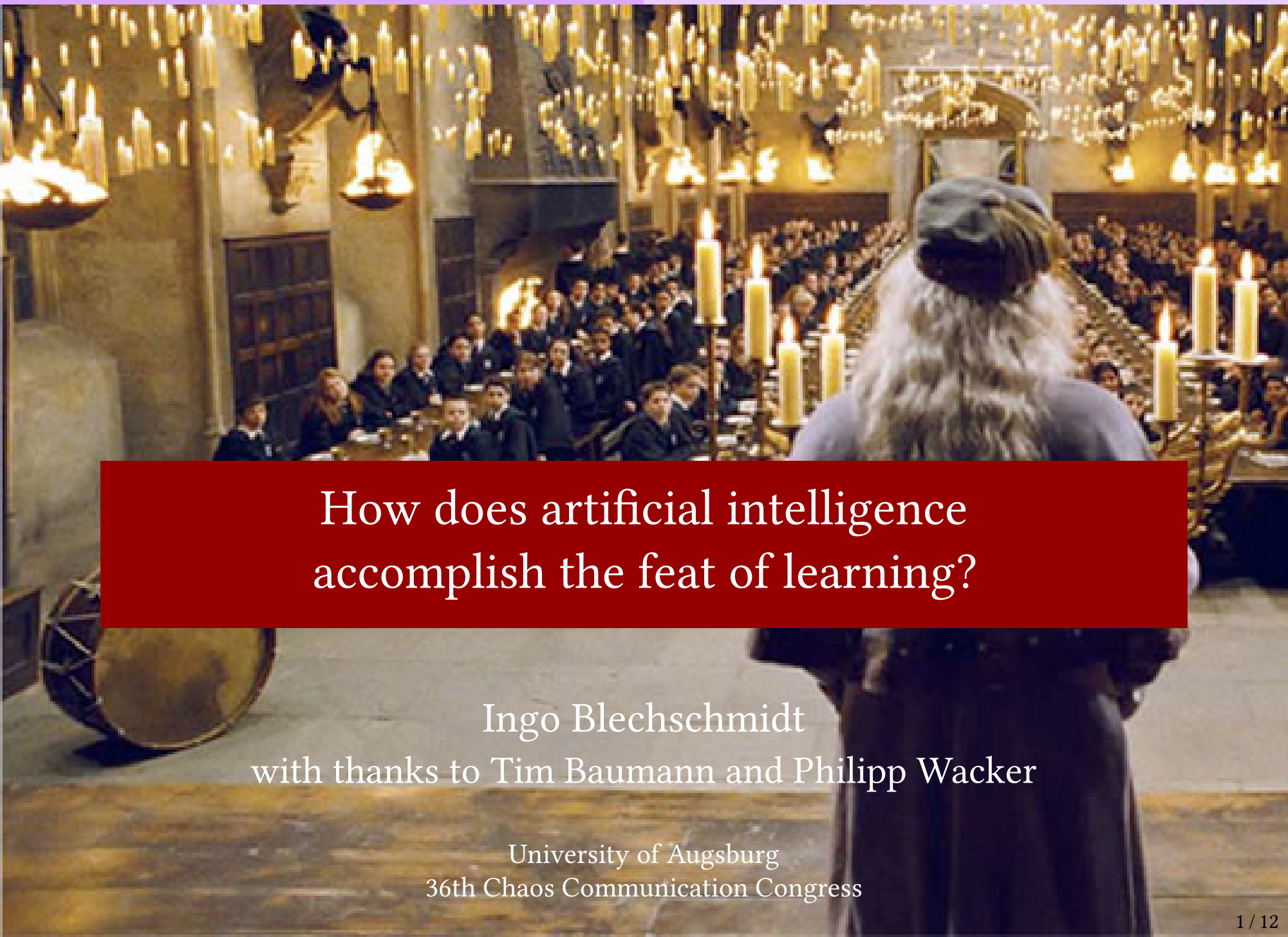


How does artificial intelligence accomplish the feat of learning?

Ingo Blechschmidt

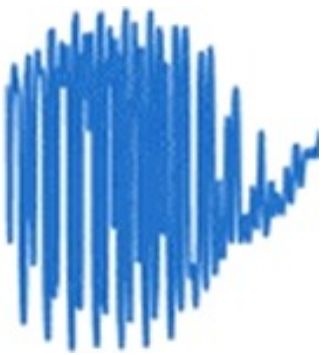
with thanks to Tim Baumann and Philipp Wacker

University of Augsburg

36th Chaos Communication Congress

Part I

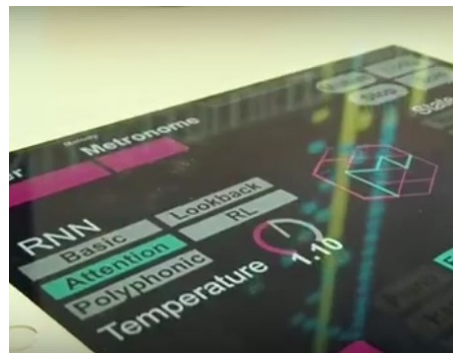
Recent successes of artificial intelligence



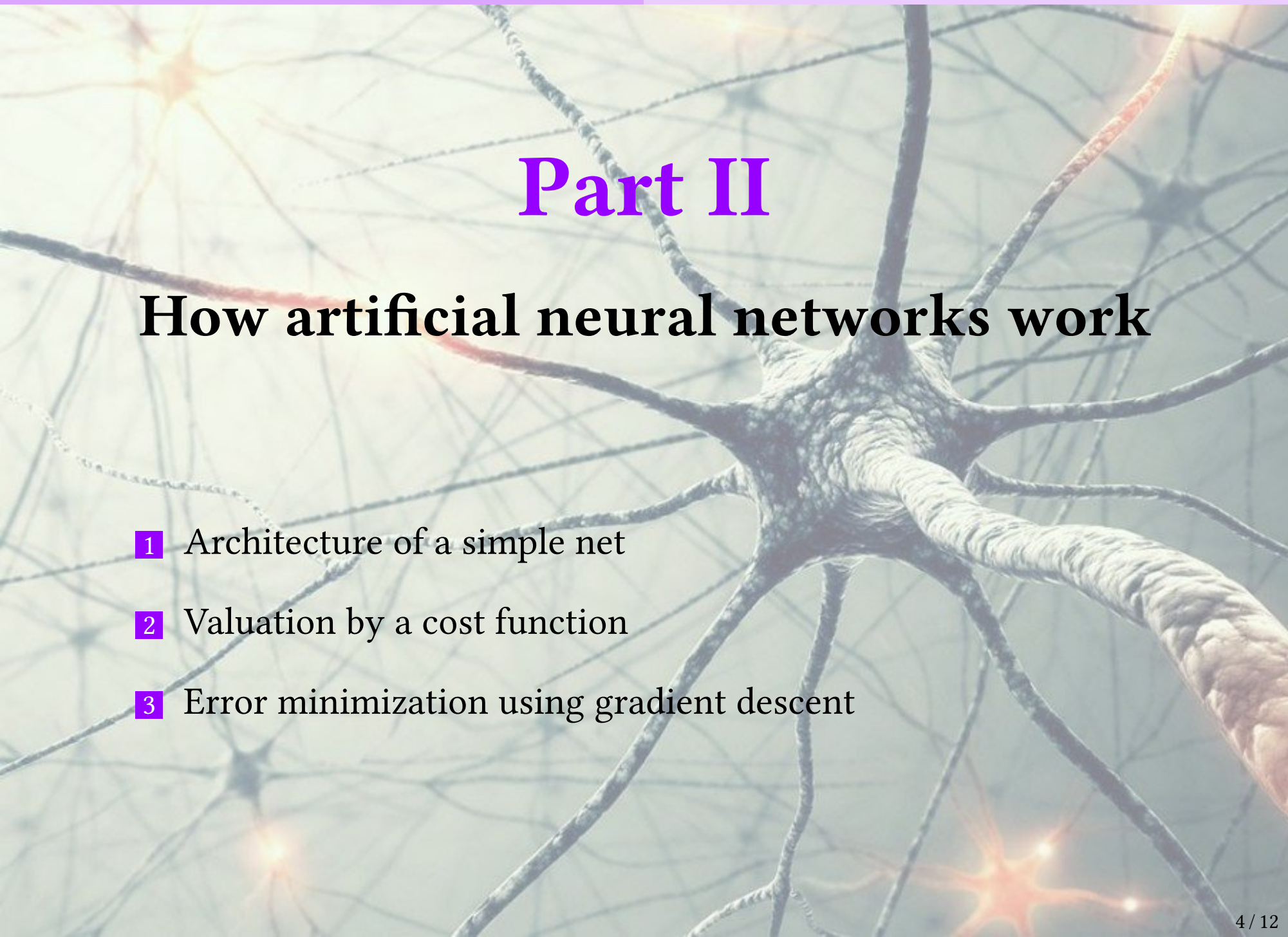
Speech synthesis

AlphaGo

Style transfer



Jam with Magenta

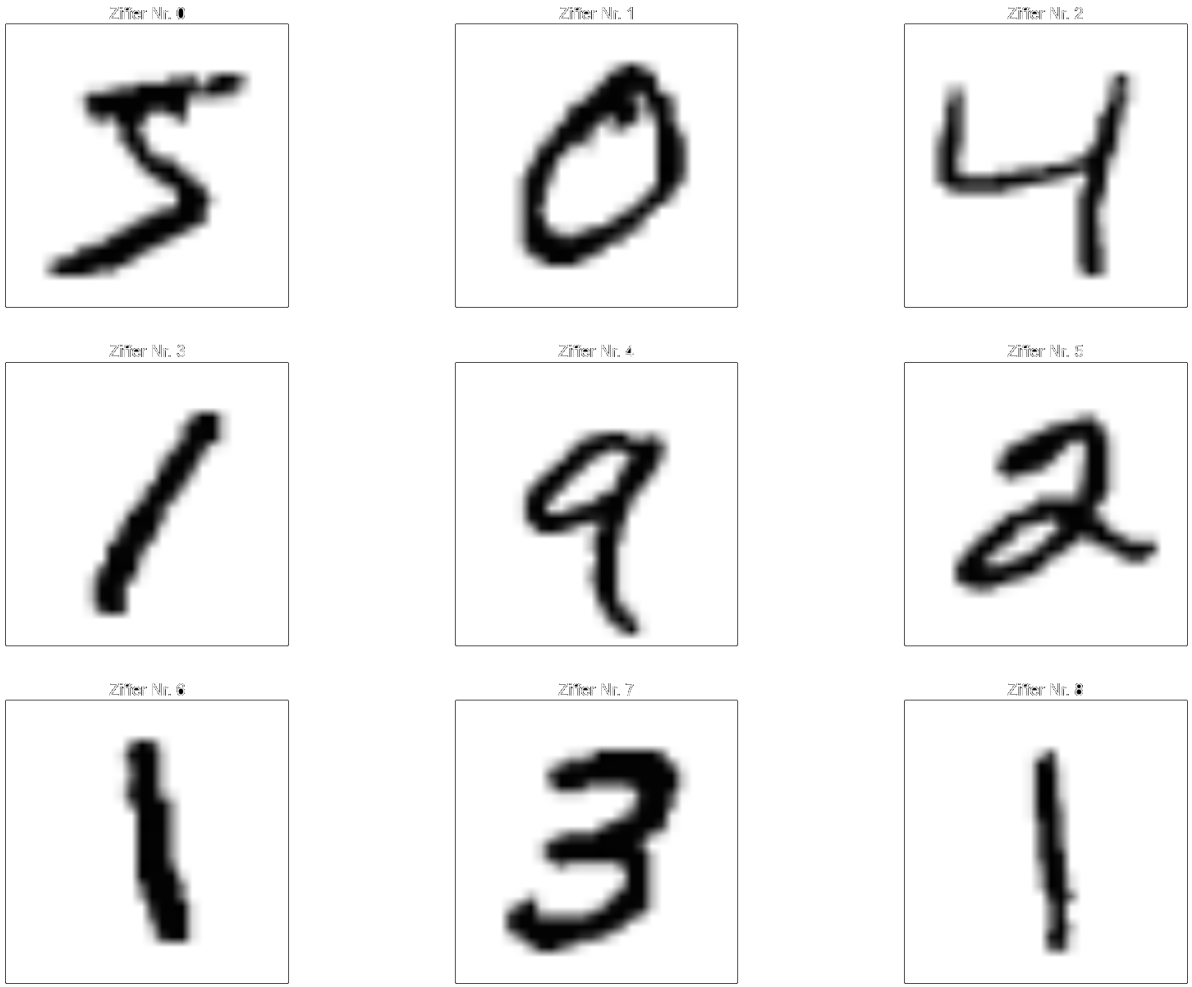


Part II

How artificial neural networks work

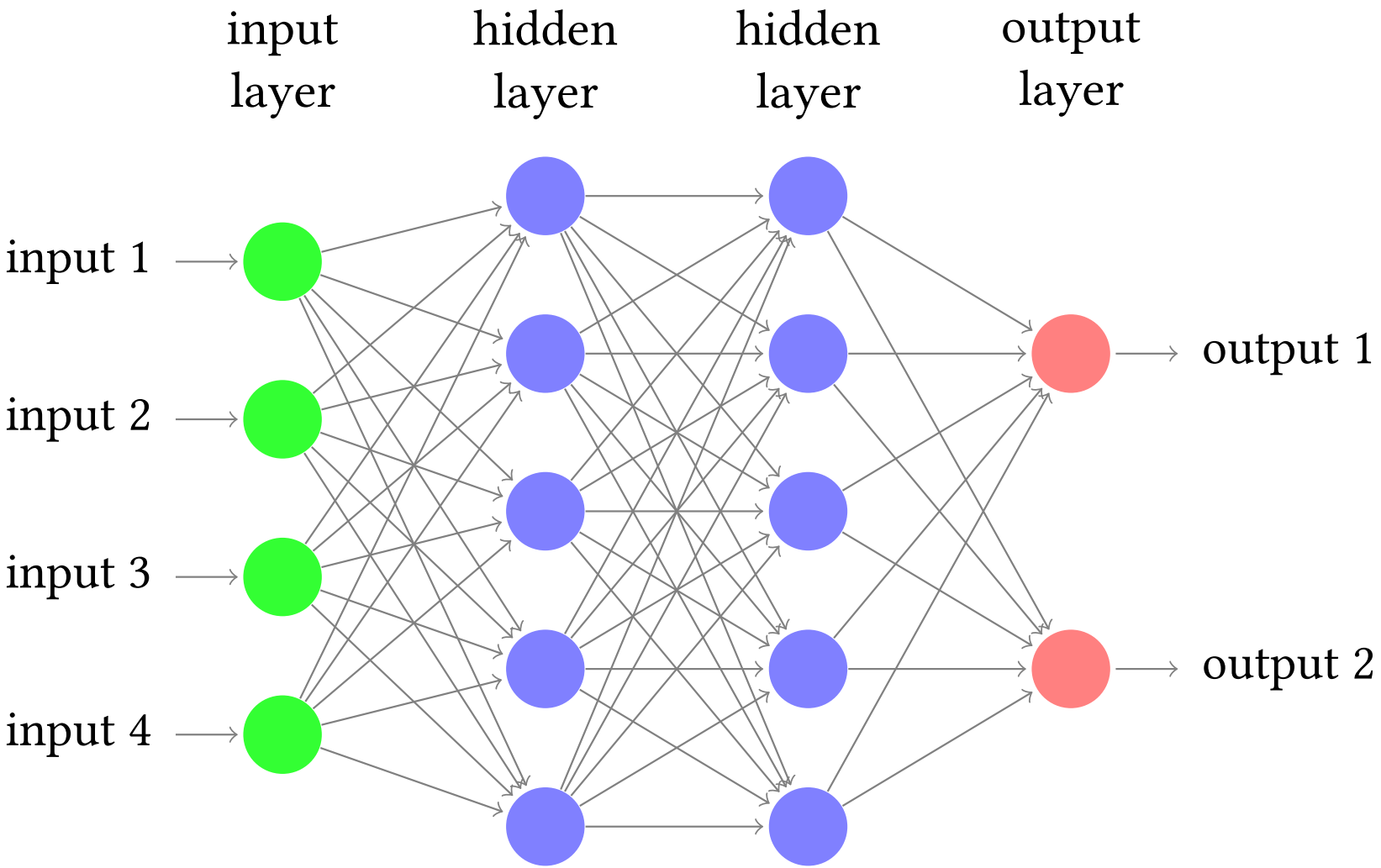
1. Architecture of a simple net
2. Valuation by a cost function
3. Error minimization using gradient descent

The MNIST database

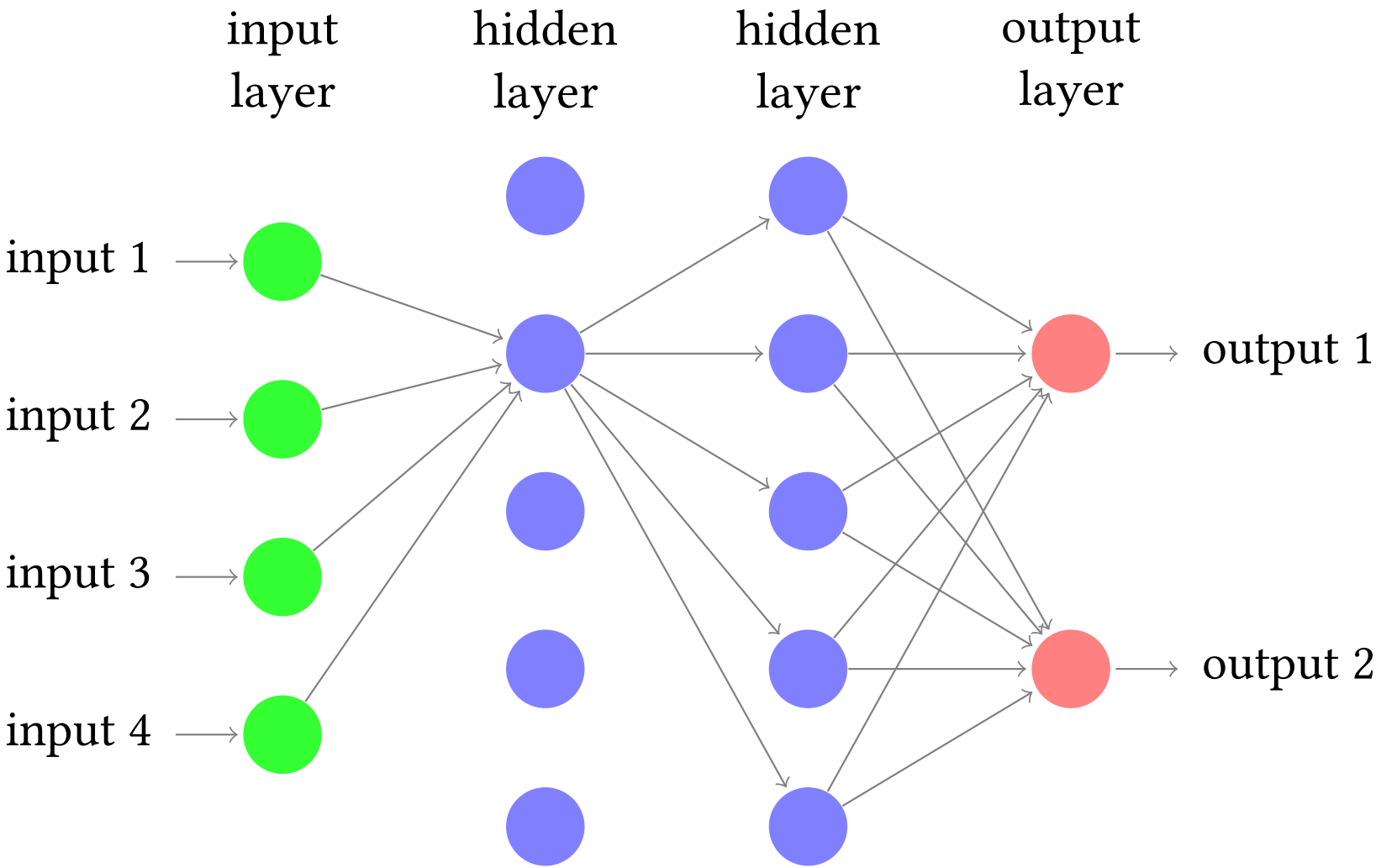


70 000 images consisting of 28×28 pixels

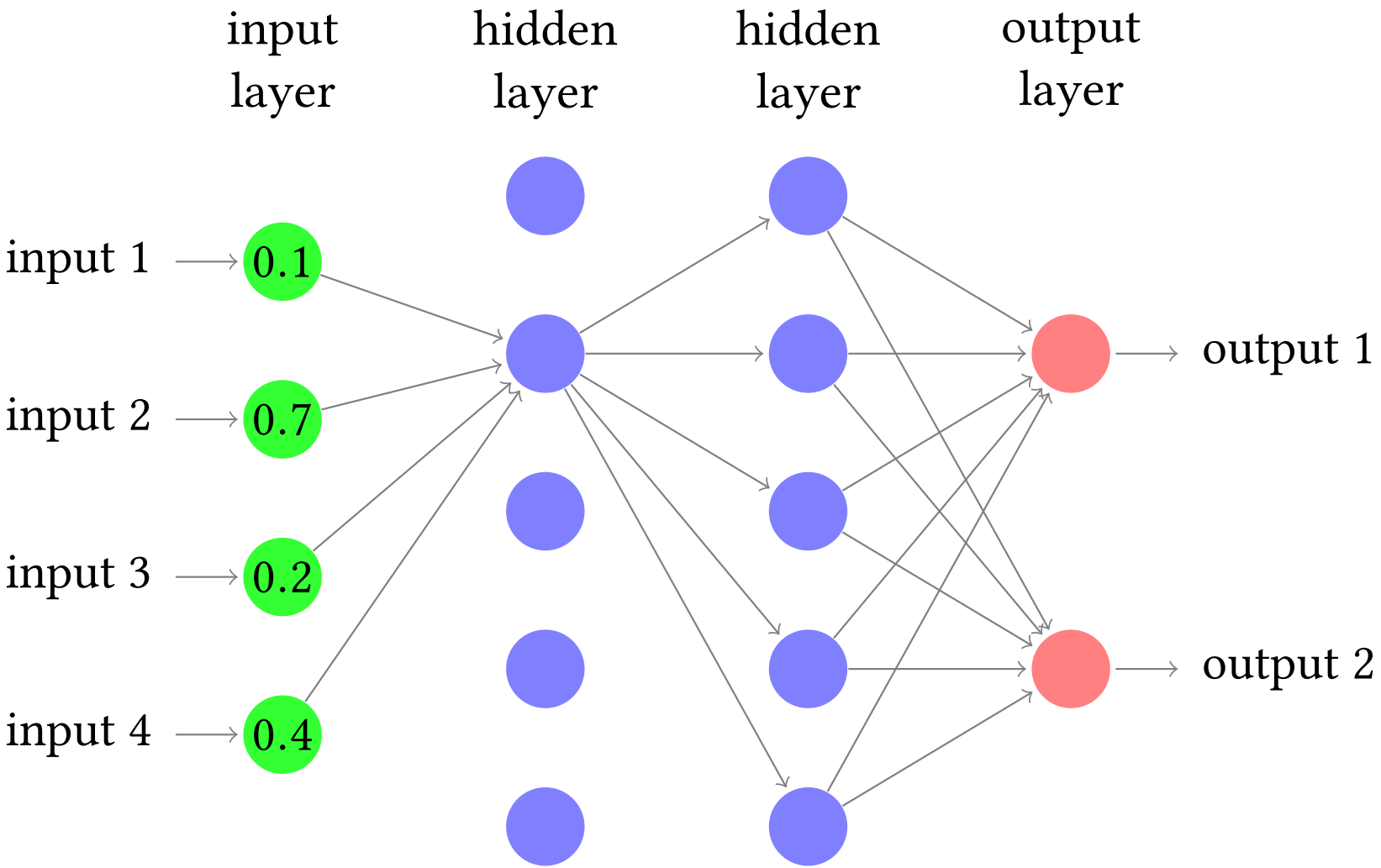
Architecture of a simple net



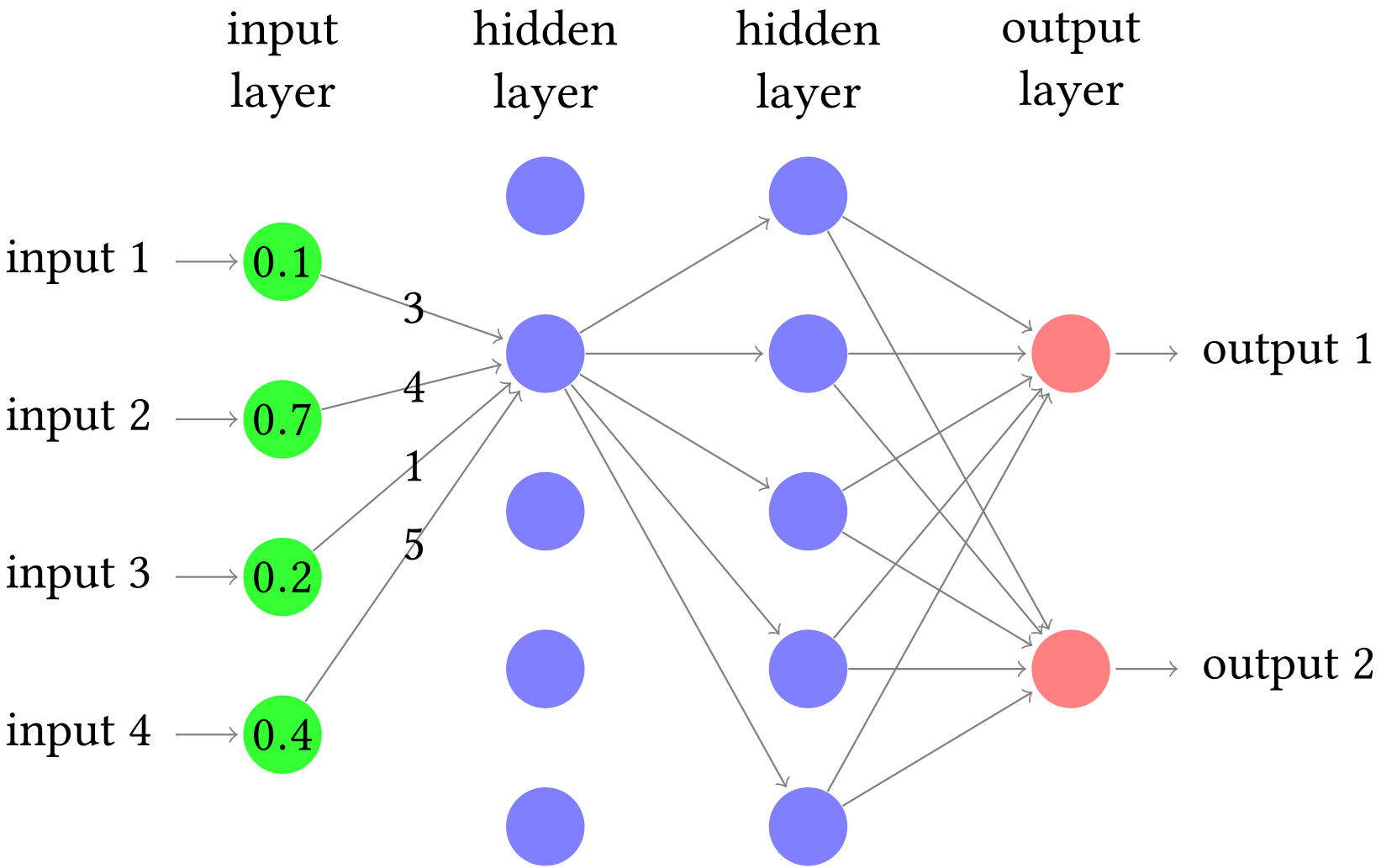
Architecture of a simple net



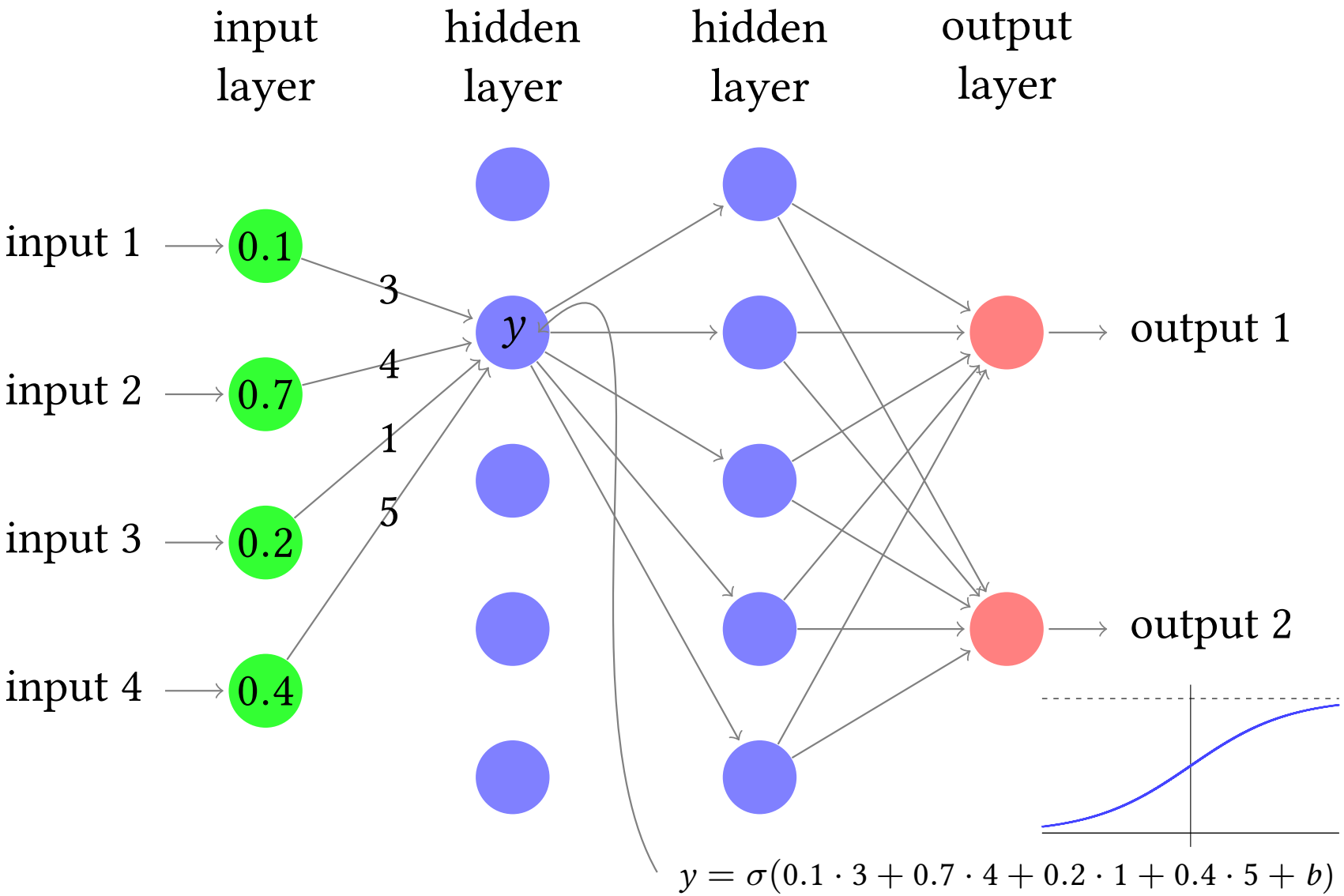
Architecture of a simple net



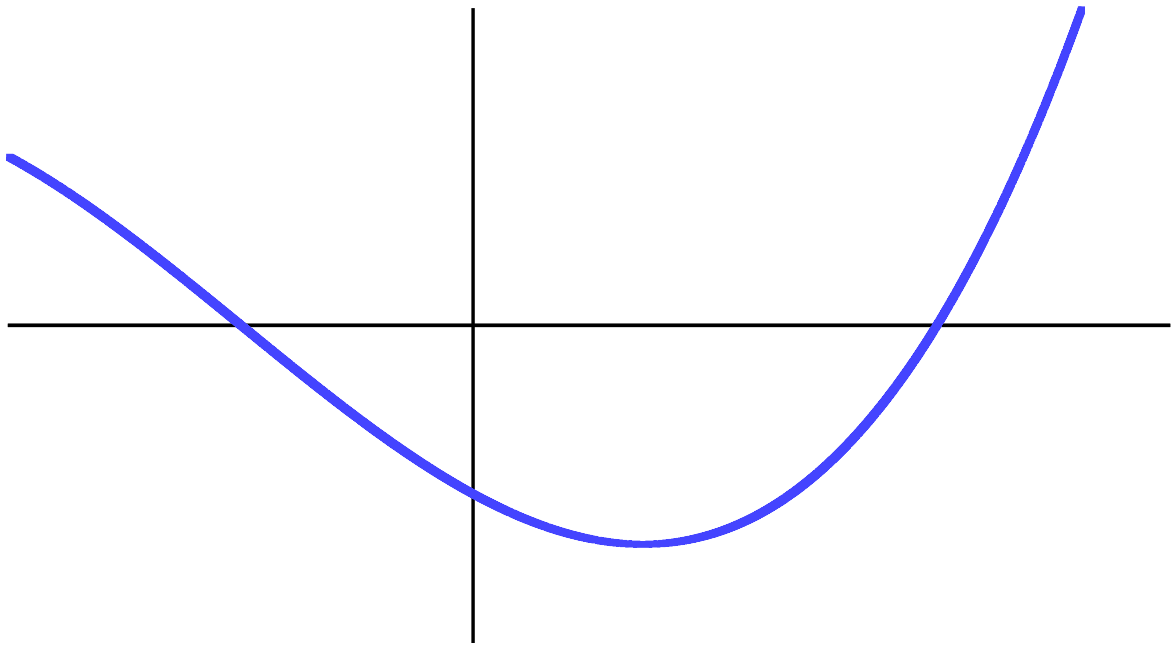
Architecture of a simple net



Architecture of a simple net

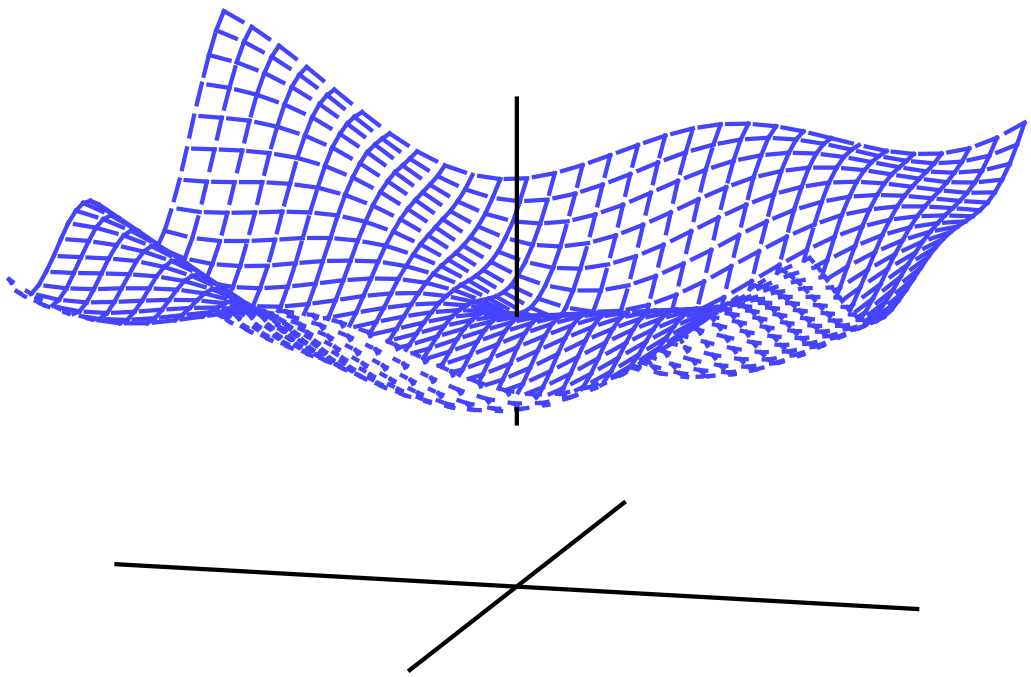


The curious importance of minimization



one unknown: x

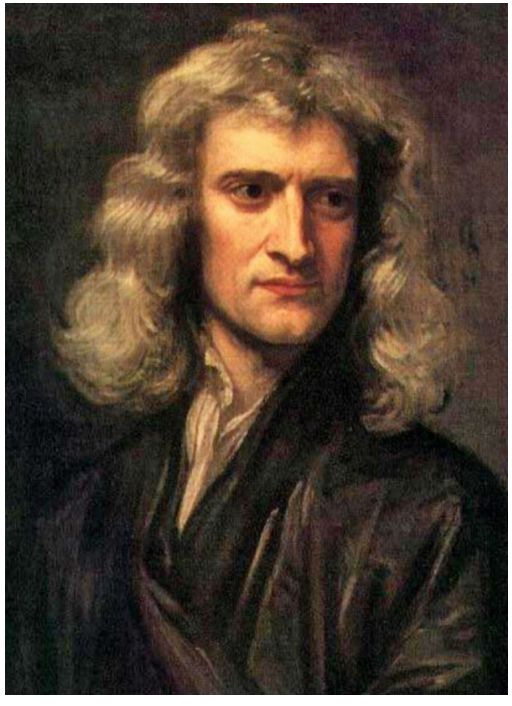
The curious importance of minimization



two unknowns: x, y

The curious importance of minimization

Leibniz (* 1646, † 1716)



Newton (* 1643, † 1727)

arbitrarily many unknowns

The feat of learning

1. Calculate for all of the 60 000 training cases the activations of the ten output neurons.
2. Sum for all of the resulting 600 000 activations the individual **quadratic errors** to obtain the **total costs**:

$$\underbrace{(0.1-0)^2+(0.7-\mathbf{1})^2+(0.1-0)^2+\cdots+(0.2-0)^2}_{\text{first test case (should be a one)}}$$

$$+\underbrace{(0.3-\mathbf{1})^2+(0.2-0)^2+(0.2-0)^2+\cdots+(0.1-0)^2}_{\text{second test case (should be a zero)}}$$

$$+\cdots$$

3. Change the weights and biases slightly in the direction of the **steepest descent** to very slightly improve performance.
4. Go to step 1.

Im MNIST-Beispiel legen wir von den 70 000 Einträgen 10 000 beiseite; diese verwenden wir später zur Validierung. Den Lernvorgang beginnen wir mit einer rein zufälligen Wahl von Gewichten und Biases.

1. Wir berechnen für jede der 60 000 Trainingsfälle die Aktivierungen der zehn Ausgabeneuronen. So erhalten wir insgesamt 600 000 Zahlen zwischen 0 und 1. Für jedes dieser Ergebnisse wissen wir, welchen Wert wir uns eigentlich wünschen (jeweils 0 oder 1 – etwa soll Ausgabeneuron Nr. 5 bei Eingabe einer handschriftlichen Sieben idealerweise überhaupt nicht feuern).

2. Für jede dieser 600 000 Ergebnisse berechnen wir den *quadratischen Fehler*

   $$(\text{tatsächliches Ergebnis} - \text{Wunschergebnis})^2$$

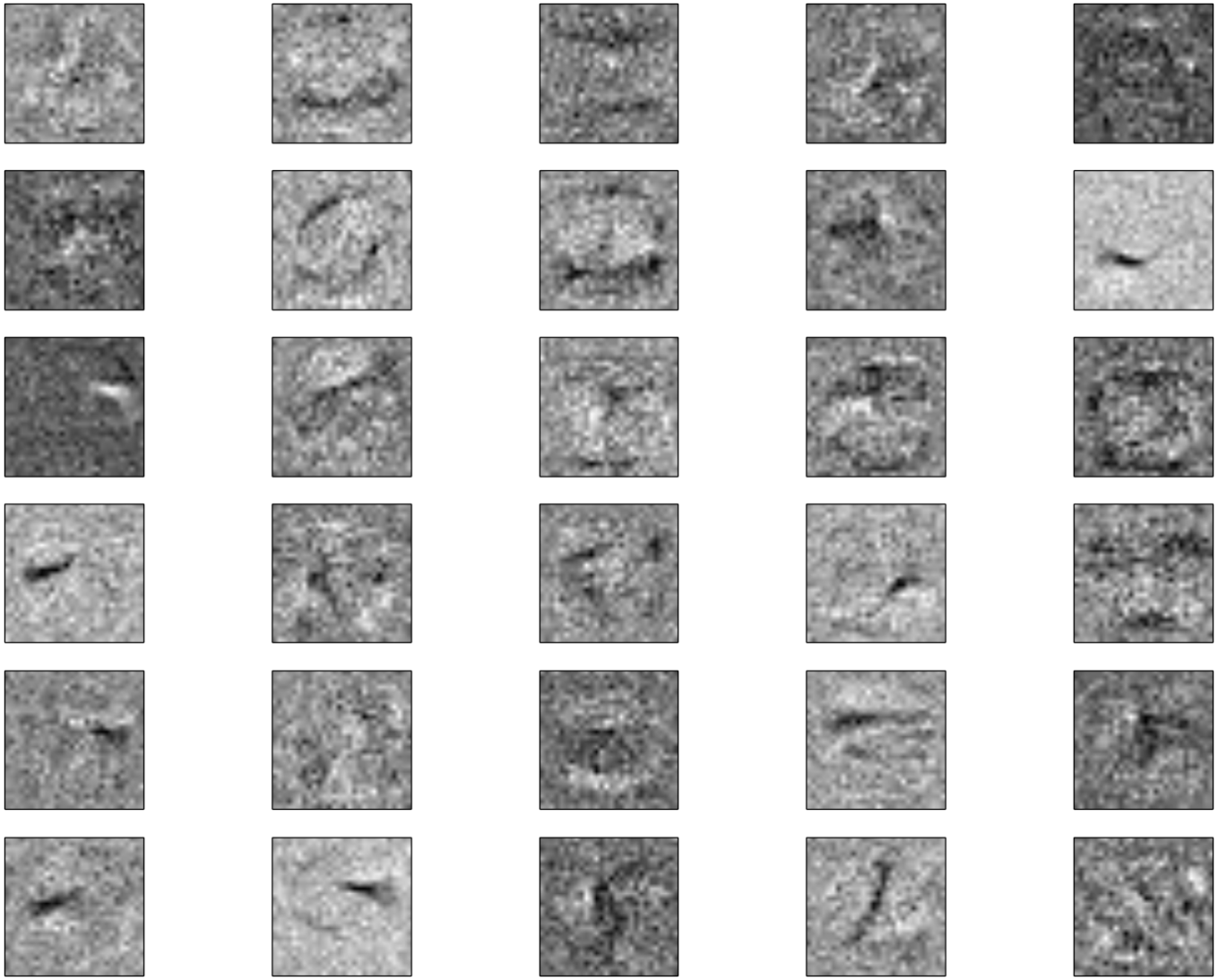
   und summieren all diese Quadrate auf. (Interessiert dich, wieso man hier quadriert? Schreibe eine Mail an iblech@speicherleck.de.)

3. Je größer diese Summe ist, desto schlechter funktioniert das Netzwerk auf den Trainingsdaten. Wir möchten daher die Summe *minimieren*. Die Summe hängt von den Gewichten der künstlichen Synapsen und den Biases der Neuronen ab; diese Abhängigkeit heißt auch *Kostenfunktion*.

4. Durch Bestimmung des *Gradienten* wissen wir, wie wir die Gewichte und Biases ändern müssen, um eine kleine Reduktion der Kostenfunktion zu erreichen. So erhalten wir neue Gewichte und Biases. Anschließend beginnen wir wieder bei Schritt 1. Auf diese Weise folgen wir zu jedem Zeitpunkt der Richtung des steilsten Abstiegs im hochdimensionalen Kostengebirge.

Sobald wir mit der Leistung des Netzes auf den Validierungsdatensätzen zufrieden sind, beenden wir das Training. Das *Wunder der Generalisierung* setzt ein: Das Netz klassifiziert auch neu geschriebene Ziffern, die nicht Teil des Trainingdatensatzes waren, sehr häufig richtig.

A look into the hidden layer



Hier wurde ein einfaches Netz zur Ziffernerkennung bestehend aus nur einer einzigen verborgenen Schicht mit 30 Neuronen trainiert. Die Grafik zeigt die Gewichte der Synapsen zwischen den 28×28 Eingabeneuronen und diesen 30 Neuronen. Das Netz hat eine Erkennungsrate von 95 %.

Verwendet man 100 Neuronen, so erreicht man 97 %; das ist fast eine Halbierung der Fehlerrate.

Demo zum Selbstprobieren:

- Python-Code zur Erkennung
- Python-Code zum Training

Part III

Why not sooner?

1. More computational power

Part III

Why not sooner?

1. More computational power
2. Availability of large data sets for training

Part III

Why not sooner?

1. More computational power
2. Availability of large data sets for training
3. Mathematical breakthrough: Convolutional Neural Networks

Part IV

Challenges for the future

- Extend neural nets to further tasks
- Understand the inner workings of a trained net
- Develop resistence against adversarial examples
- Solve ethical challenges with self-driving cars
- Answer existential questions regarding strong AI

Wie ein künstliches neuronales Netzwerk funktioniert, ist – anders als bei herkömmlichem Programmcode – nicht klar. (Wie beim Menschen auch.) Dazu wird momentan aktiv geforscht. Zwei Einstiegspunkte zu solchen Untersuchungen sind:

- Inceptionism: Going Deeper into Neural Networks von Alexander Mordvintsev, Christopher Olah und Mike Tyka
- Visualizing and Understanding Convolutional Networks von Matthew Zeiler und Rob Fergus

Part V

Recommendations

- HBO series *Westworld* about androids who pass the Turing test and develop consciousness
- Talks by Joscha Bach on previous congresses
- The Unreasonable Effectiveness of Recurrent Neural Networks by Andrej Karpathy
- TensorFlow – AI development without prerequisites in maths
- Neural Networks and Deep Learning by Michael Nielsen