Deep Learning on SpiNNaker

Jonas Fassbender
jonas@fassbender.dev

THE UNIVERSITY *of* EDINBURGH

Supervisors:
Alan Stokes, Kevin Stratford, Caoimhín Laoide-Kemp

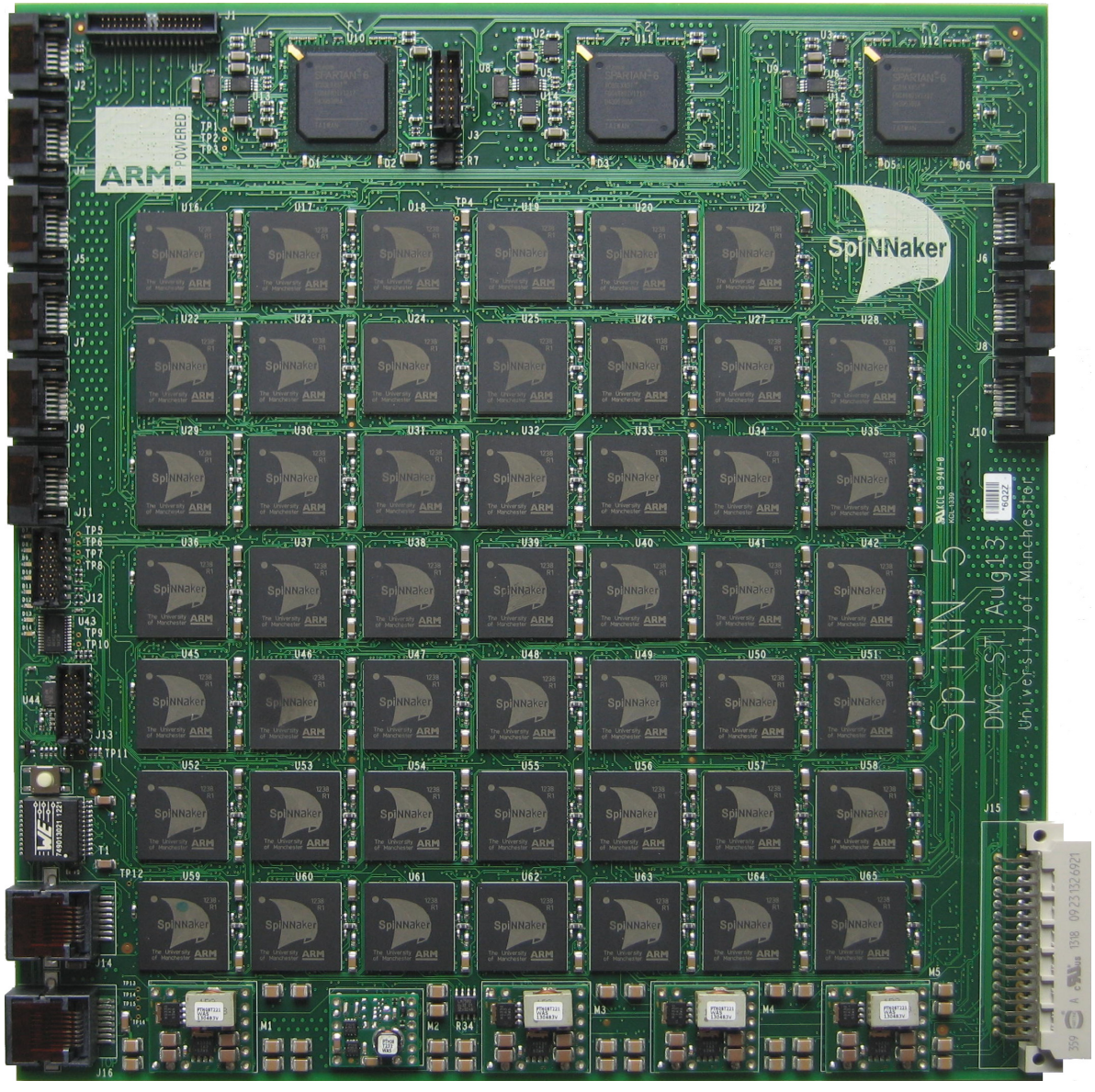
epcc

SpiNNaker

SpiNNaker

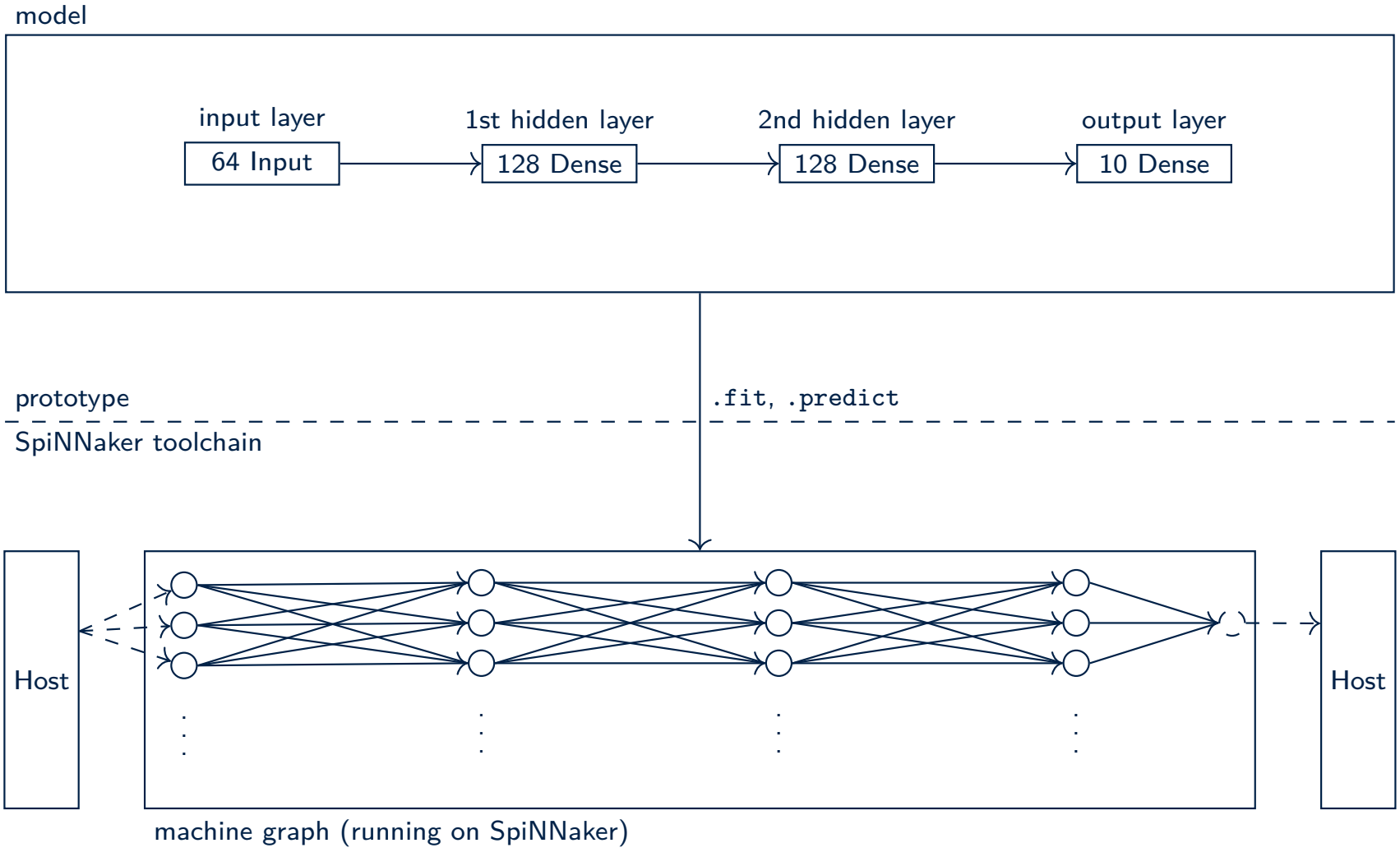
[SpiNNaker is] a platform for high-performance massively parallel [and energy efficient] processing appropriate for the simulation of large-scale [spiking] neural networks [6]

SpiNNaker


ARM POWERED
SpiNNaker
SpiNN-5
DMG ST Aug13
University of Manchester

Prototype

Prototype Architecture



Features

Supported Features

- Dense layers
- 1D convolutional layers
- (Flatten layers)
- Most common activation functions
- Most common loss functions
- SGD optimizer with constant learning rate

Features

Supported Features

- Dense layers
- 1D convolutional layers
- (Flatten layers)
- Most common activation functions
- Most common loss functions
- SGD optimizer with constant learning rate

Missing Features

- 2D convolutional layers
- Pooling layers
- Shortcut connections [5]
- Batch normalization

Problems

Problems

- TIME!

Problems

- TIME!
- Dropped packets

Problems

- TIME!
- Dropped packets
- Convolutional layer as neurons is a complex abstraction

Problems

- TIME!
- Dropped packets
- Convolutional layer as neurons is a complex abstraction
- Issues with the toolchain [1, 4, 2, 3]

Next Steps

Overcoming Dropped Packets in the Future

SpiNNaker APIs we were not able to try:

Overcoming Dropped Packets in the Future

SpiNNaker APIs we were not able to try:

- Application graph instead of machine graph (less packets sent)

Overcoming Dropped Packets in the Future

SpiNNaker APIs we were not able to try:

- Application graph instead of machine graph (less packets sent)
- Slower communication protocol for bigger messages

Overcoming Dropped Packets in the Future

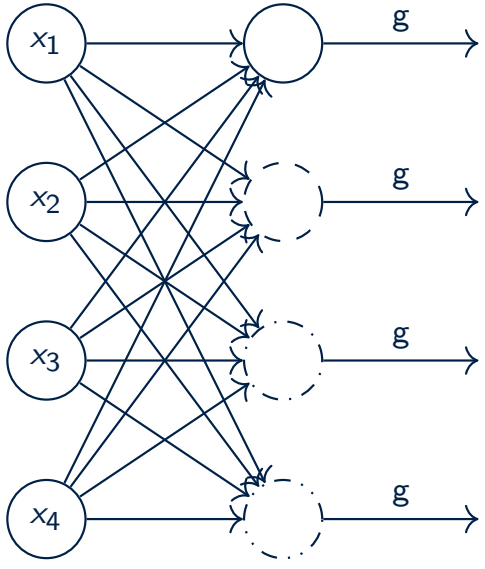
SpiNNaker APIs we were not able to try:

- Application graph instead of machine graph (less packets sent)
- Slower communication protocol for bigger messages
- Time-division multiple access (TDMA) system (sends packets at the right time to avoid dropping them)

Overcoming Dropped Packets in the Future

Better domain decomposition:

$$\begin{pmatrix} \boxed{w_{11}} & \boxed{w_{12}} & \boxed{w_{13}} & \boxed{w_{14}} \\ \boxed{w_{21}} & \boxed{w_{22}} & \boxed{w_{23}} & \boxed{w_{24}} \\ \boxed{w_{31}} & \boxed{w_{32}} & \boxed{w_{33}} & \boxed{w_{34}} \\ \boxed{w_{41}} & \boxed{w_{42}} & \boxed{w_{43}} & \boxed{w_{44}} \end{pmatrix}^{\top} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} \boxed{b_1} \\ \boxed{b_2} \\ \boxed{b_3} \\ \boxed{b_4} \end{pmatrix}$$



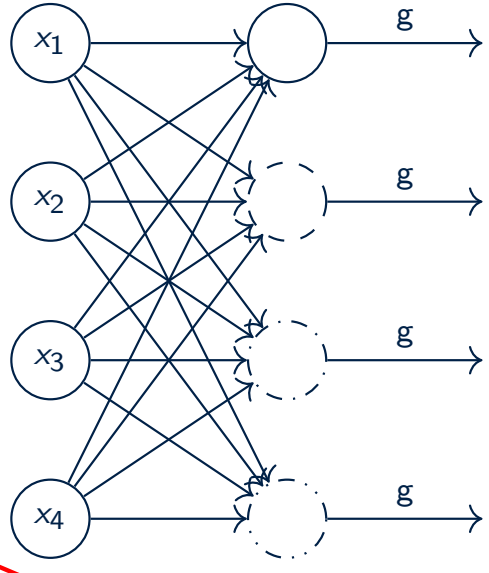
number of packets sent in each direction: 16 → 64 ←

Figure: How the computational graph of a dense layer is decomposed into neurons (implemented by the prototype).

Overcoming Dropped Packets in the Future

Better domain decomposition:

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{pmatrix}^{\top} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$



Number of vertices · number of edges · number of sends per vertex

number of packets sent in each direction:

16 →

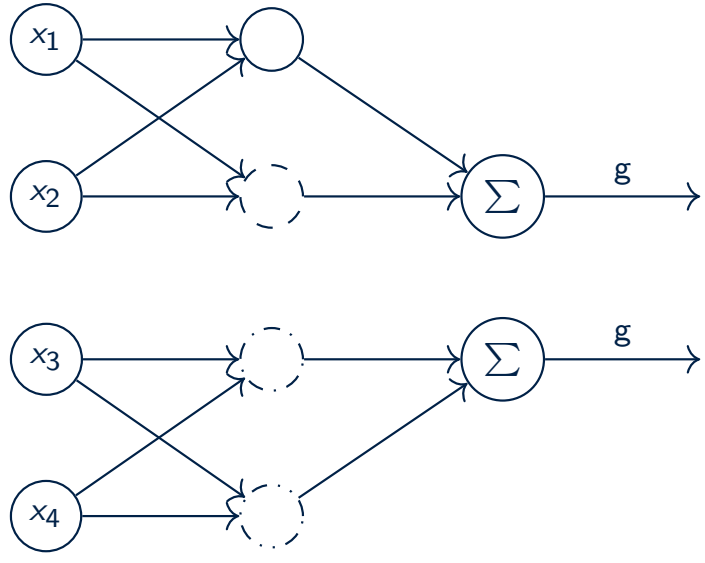
64 ←

Figure: How the computational graph of a dense layer is decomposed into neurons (implemented by the prototype).

Overcoming Dropped Packets in the Future

Better domain decomposition:

$$\begin{pmatrix} \boxed{w_{11} & w_{12}} & w_{13} & w_{14} \\ \boxed{w_{21} & w_{22}} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{pmatrix}^{\top} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$



number of packets sent in each direction:

8 → 32 ← 8 → 8 ←

Figure: How the computational graph of a dense layer could be decomposed in order to reduce the peak and overall number of packets by sparsifying the graph.

Summary

Summary

- Unfortunately no ResNet-50

Summary

- Unfortunately no ResNet-50
- Valuable insights gained

Summary

- Unfortunately no ResNet-50
- Valuable insights gained
- Learned a lot (SpiNNaker, deep learning, software engineering, project management, ...)

Code Example


```
import numpy as np
import tensorflow as tf

# generate random data set
X = np.random.rand(500, 10)

# Keras MLP
kmodel = tf.keras.models.Sequential()
kmodel.add(tf.keras.layers.Dense(
  50, activation="relu",
  input_shape=(10,)))
kmodel.add(tf.keras.layers.Dense(
  50, activation="softmax"))
kmodel.add(tf.keras.layers.Dense(
  300, activation="tanh"))
kmodel.add(tf.keras.layers.Dense(
  50, activation="sigmoid"))
kmodel.add(tf.keras.layers.Dense(25))

# predict labels from random data
# with random weights
p_ = kmodel.predict(X)
```

```
from spiDNN import Model
from spiDNN.layers import Input, Dense

# equivalent model for SpiNNaker
model=Model()
model.add(Input(10))
model.add(Dense(50, activation="relu"))
model.add(Dense(50, activation="softmax"))
model.add(Dense(300, activation="tanh"))
model.add(Dense(50, activation="sigmoid"))
model.add(Dense(25))

# make sure both models have same parameters
model.set_weights(kmodel.get_weights())

p = model.predict(X)

# floating point discrepancies may occur
assert np.amax(np.absolute(p - p_)) < 1e-4
```

References

[1] J. Fassbender. Encountered an issue within the live_packet_gather.c file (flush_events seems to corrupt the circular_buffer 'with_payload_buffer'), 2020.

[2] J. Fassbender. fixed wrong offset into key_array, 2020.

[3] J. Fassbender. "OSError: [Errno 98] Address already in use" when trying to run a second LiveEventConnection in the same program (after the first was closed properly), 2020.

[4] J. Fassbender. 'ReverseIPTagMulticastSourceMachineVertex' was missing its 'n_keys' argument which I need, because I work with a machine graph and not an application graph, 2020.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] SpiNNaker. SpiNNaker Project. `http://apt.cs.manchester.ac.uk/projects/SpiNNaker/project/`, 2020.

epcc