



High-Fidelity Conversion of Floating-Point Networks for Low-Precision Inference Using Distillation with Limited Data

James Imber
Imagination Technologies

1. Problem definition
2. Training for low-precision inference
3. Distillation and label-free training
4. Universal cost function
5. Dataset security
6. Results

Despite increasing interest in Quantization Aware Training (QAT), most training is still done in 32-bit floating point.

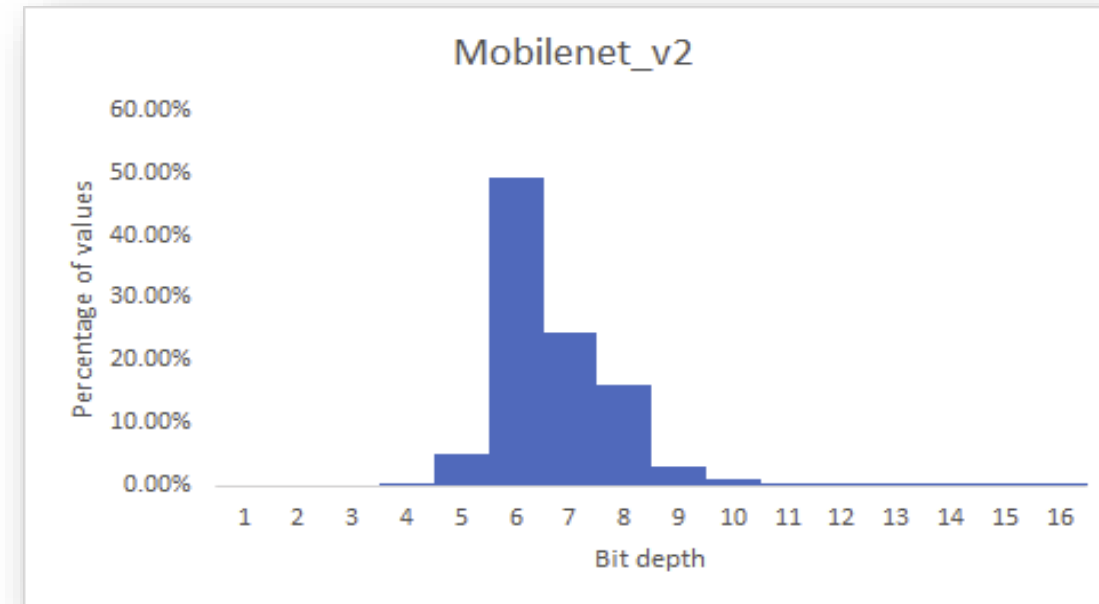
For deployment, maximum efficiency is preferable:

- low bandwidth
- low power
- high compute density
- low area

Many NNA designs support low bit depth inference for this reason.

This presentation describes methods for converting 32-bit floating point networks to low bit depths which:

- use minimal amounts of unlabelled data
- maintain high fidelity to the original network even at low bit depths.

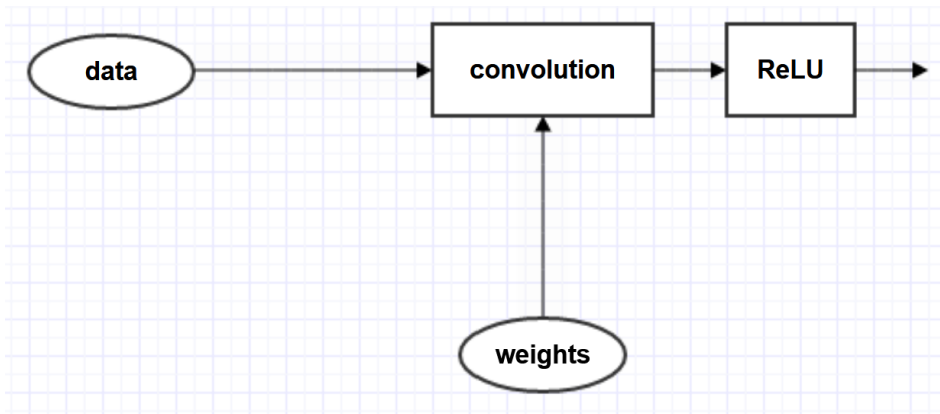


Training for Low-Precision Inference

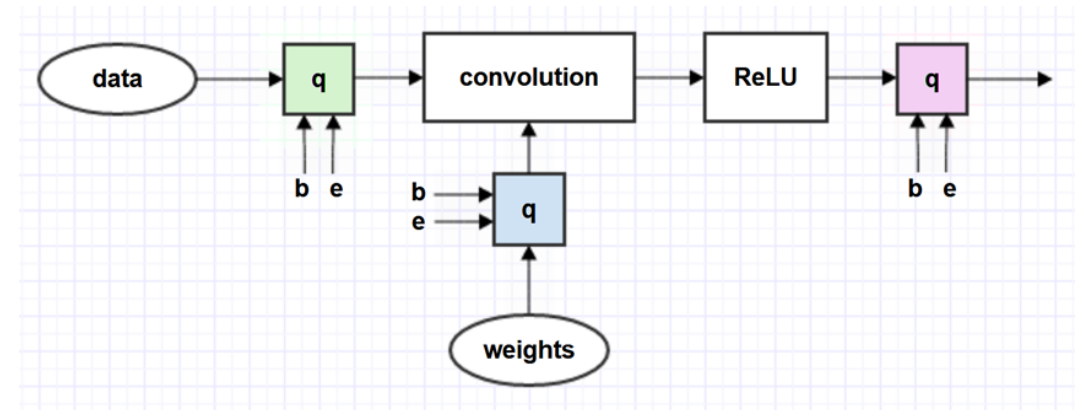
Number formats can be represented by one or more *quantization parameters*. We want formats with as few bits as possible without damaging accuracy.

For example, a shared exponent and number of bits can be given for every tensor or channel.

Before quantisation:



After quantisation:

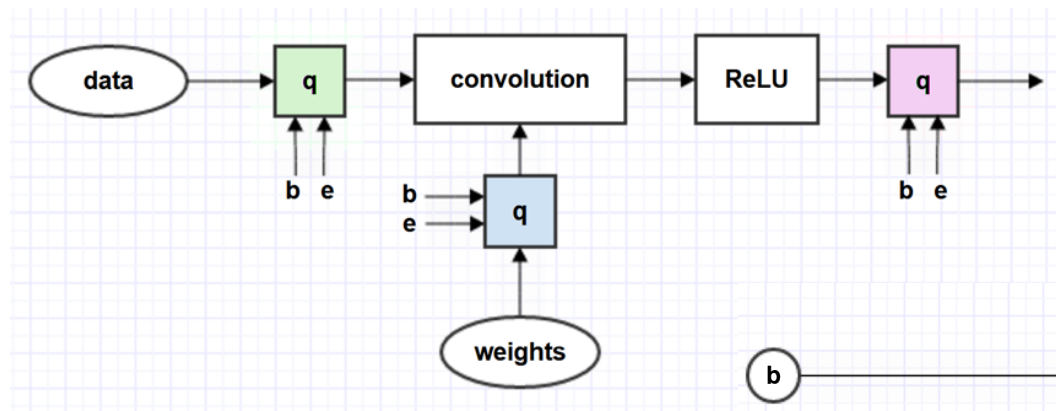


The standard way to do this in QAT is to insert “fake” quantization nodes that match the target hardware.

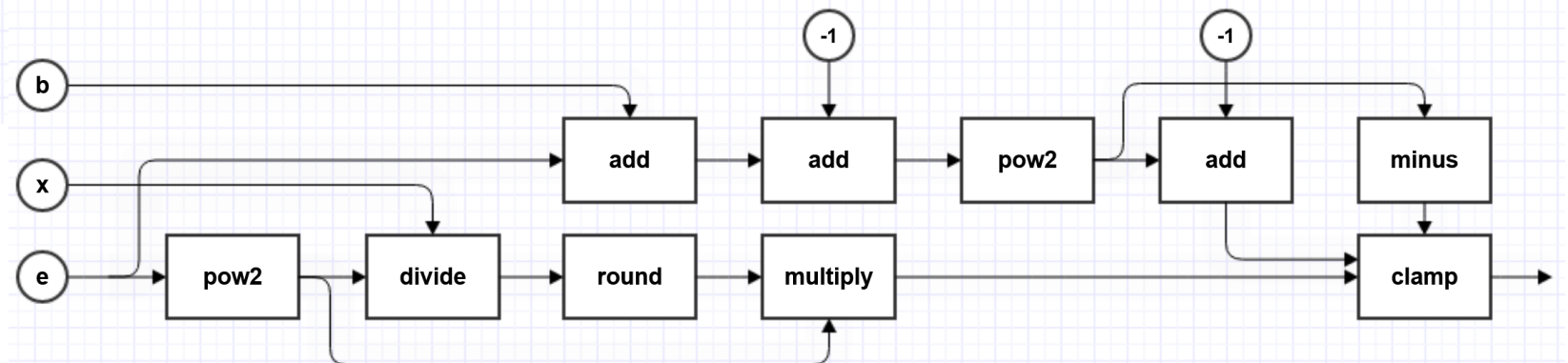
Training for Low-Precision Inference

Our fake quantisation nodes are designed so that we can backpropagate gradients to the quantization parameters.

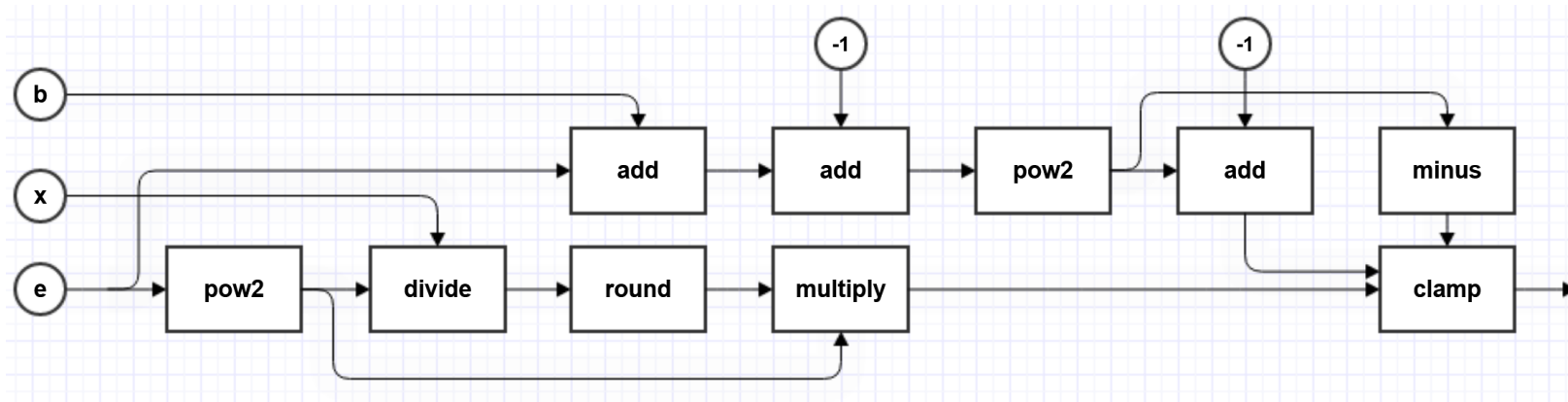
This lets us learn suitable quantization parameters during training that minimize number of bits while maximizing accuracy.



Quantisation node q:



Training for Low-Precision Inference



Building our fake quantization nodes out of math ops makes implementation simpler.

On the backward pass, everything is differentiable except the rounding op.

We override the gradient for the round op to a constant 1 – the “straight-through” estimator. (Hubara et al. [1] following earlier work by Hinton and Bengio).

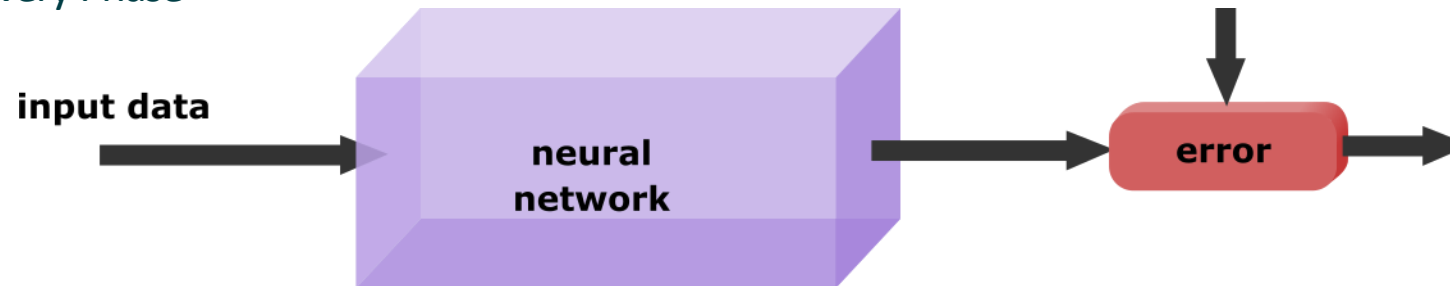
[1] *Binarized Neural Networks*

Hubara, Courbariaux, Soudry, El-Yaniv and Bengio

NIPS 2016

Originally proposed by Hinton et al. [2] for transferring a learned mapping to smaller models.
The original network is used for discovering the mapping, and the smaller one uses the output as a *soft target*.

Step 1: Discovery Phase



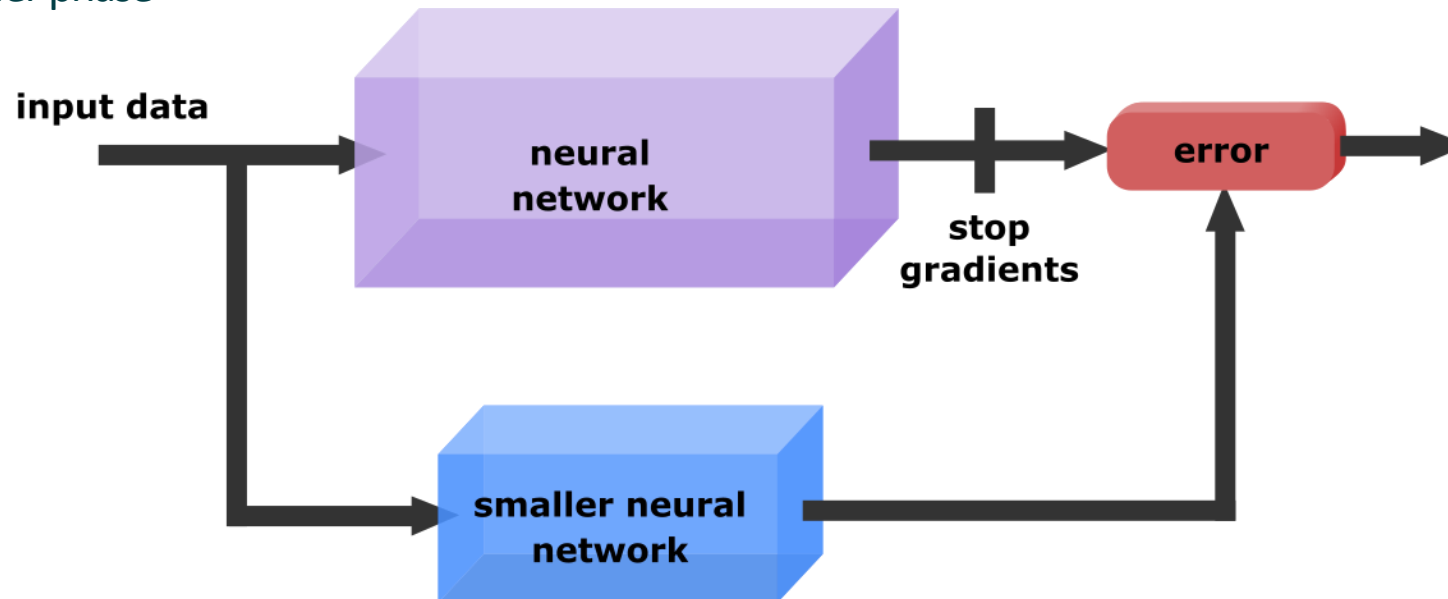
[2] *Distilling the Knowledge in a Neural Network*

Hinton, Vinyals and Dean

NIPS Deep Learning and Representation Learning Workshop (2015)

Originally proposed by Hinton et al. [2] for transferring a learned mapping to smaller models. The original network is used for discovering the mapping, and the smaller one uses the output as a *soft target*.

Step 2: Transfer phase



[2] *Distilling the Knowledge in a Neural Network*

Hinton, Vinyals and Dean

NIPS Deep Learning and Representation Learning Workshop (2015)

There are a couple of very interesting things to note about the second step:

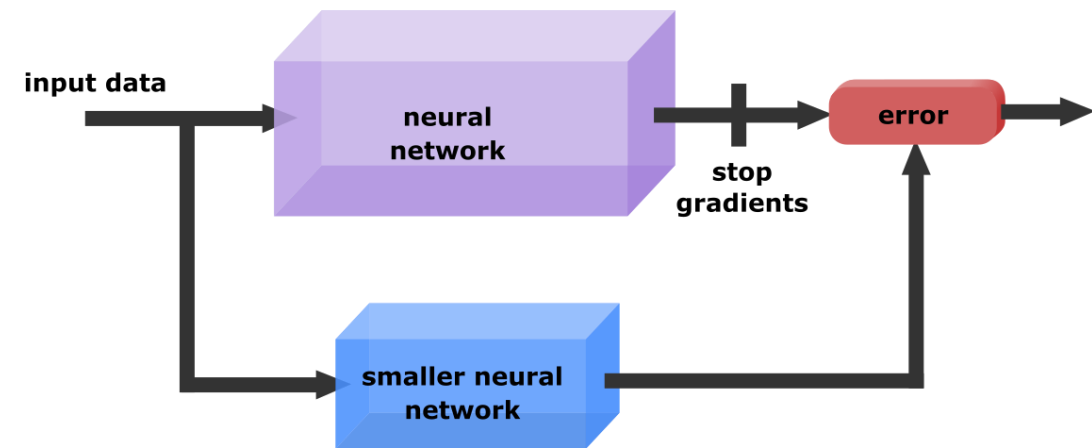
There are no labels!

Only need input data and the original network.

It's very general.

You can define cost functions that work for practically any supervised learning task mapping inputs to outputs.

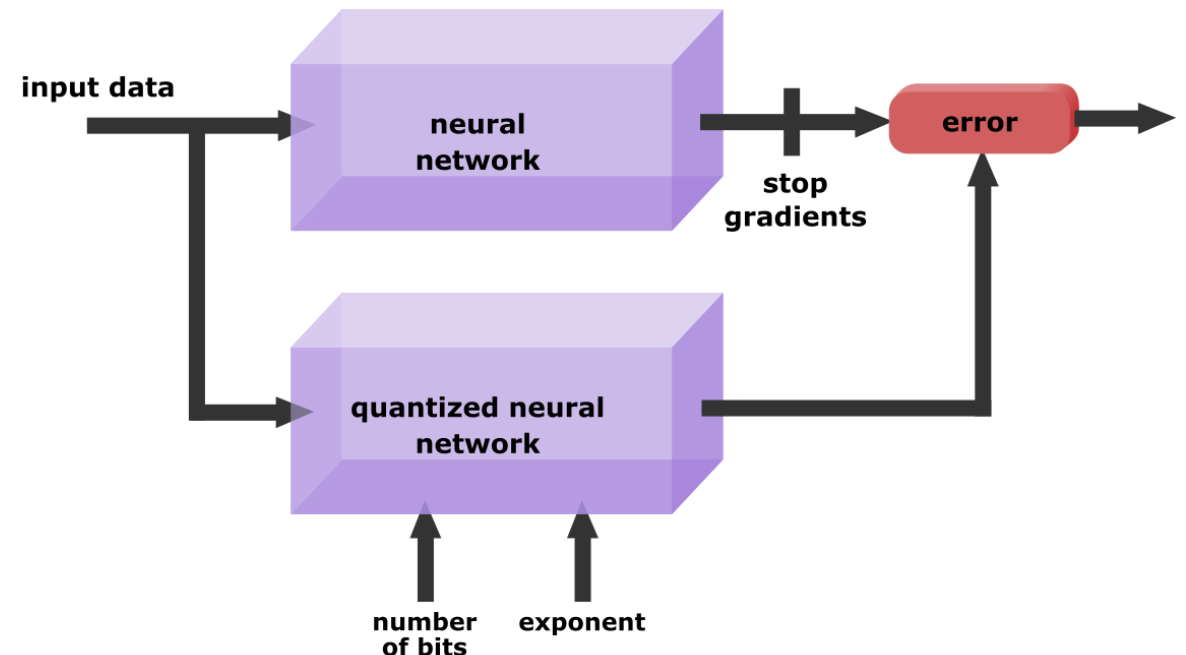
We found that these properties are very useful in practice for preparing networks for low-precision inference.



Distillation can be used to train the quantization parameters of our quantized network.

Gradients can be backpropagated to the **number of bits** and **exponent**, and used for training in a DL framework (e.g. PyTorch).

I've only shown one pair in the diagram for simplicity, but we train all the number formats simultaneously.



Universal Cost Function

Instead of using a problem-specific loss function, we use a universal one.

The loss function we minimise is:

$$|f(x) - f_q(x)| + \gamma B$$

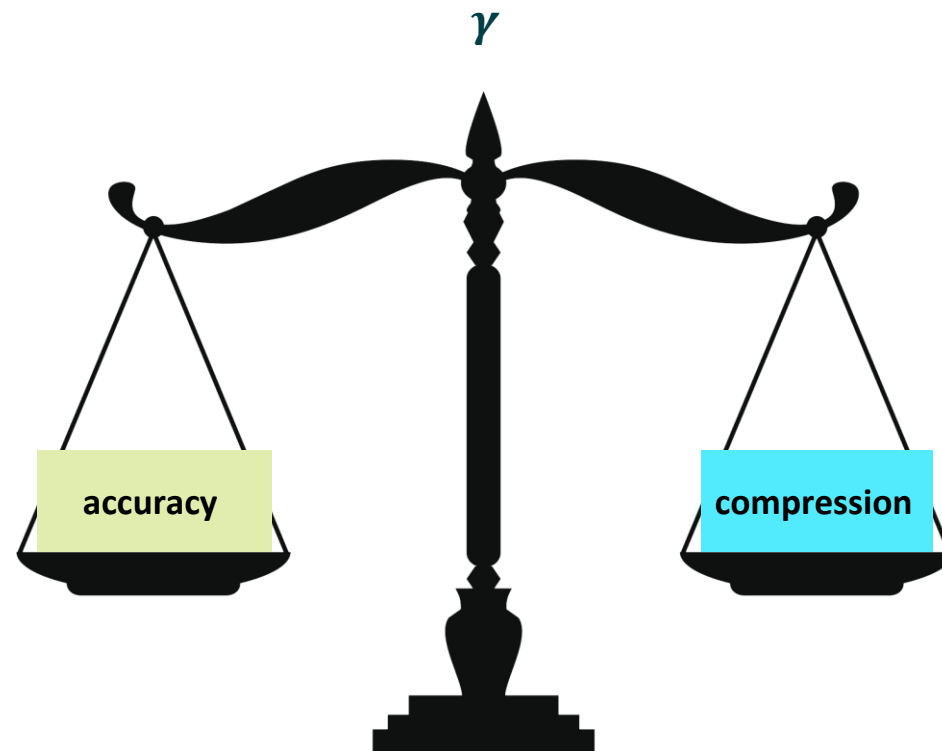
$|f(x) - f_q(x)|$ is the *distillation loss*: the difference between the output of the original and the quantised network.

We use an L1 norm – this works well for everything from image classification to style transfer!

B is the average number of bits in the network.

γ is used to balance between compression and accuracy. This is a hyperparameter that needs to be set by the user.

Some care needs to be taken with the learning rate during training.



Universal Cost Function

Instead of using a problem-specific loss function, we use a universal one.

The loss function we minimise is:

$$|f(x) - f_q(x)| + \gamma B$$

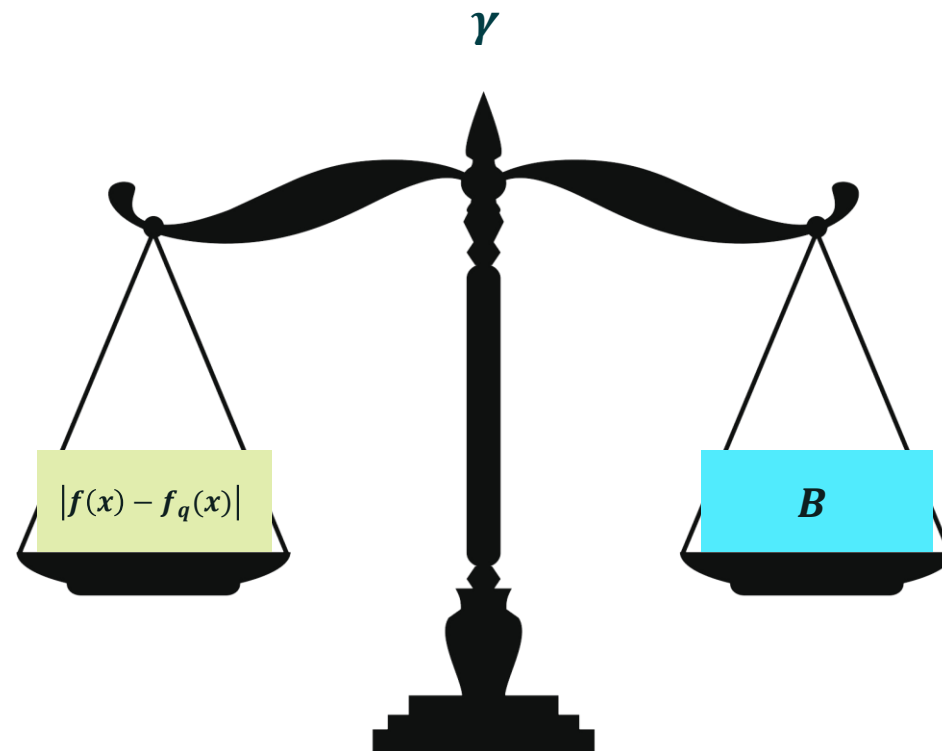
$|f(x) - f_q(x)|$ is the *distillation loss*: the difference between the output of the original and the quantised network.

We use an L1 norm – this works well for everything from image classification to style transfer!

B is the average number of bits in the network.

γ is used to balance between compression and accuracy. This is a hyperparameter that needs to be set by the user.

Some care needs to be taken with the learning rate during training.



Data would ordinarily need to be shared with a third party performing model compression.

Our proposed distillation-based compression scheme means that large datasets (and in particular labels) **need not be shared**.

COMPLIANCE RISK

Many jurisdictions have laws restricting the collection and sharing of personal data.

Datasets containing personal data may have to be anonymized before they can be shared – this can be costly and time-consuming.

OPERATIONAL RISK

Datasets often represent a significant investment by an organisation. They are a closely-guarded asset.

Most companies are (understandably!) protective of their datasets.

Labels represent a large part of the value of many datasets.

Advantages of Distillation for Low-Precision Inference

LABEL-FREE TRAINING

Labels are **valuable**. Companies take a risk whenever they share them externally.

Labels may contain **sensitive data**. Distillation helps ensure privacy.

GENERALITY

Simple, reusable compression code.

The same **general-purpose cost function** can be used for a very wide variety of tasks.

LIMITED DATA

No need for access to **large, proprietary datasets** since we are learning very few parameters.

A small number of representative inputs is all that's needed.

SOFT TARGETS

Easier and faster to learn **formats** with distillation loss than with the loss used to train the original, floating point network.

HIGH FIDELITY

We are **training to match the original network**, not the dataset. This results in a closer match to the original network.

WEIGHT FINE-TUNING

We can **fine-tune weights jointly** with learning the formats for improved accuracy.

However, this requires more data.



Results

This method can be applied to virtually any low-precision number format.

We have tested it with QNA (for learned N , as an extension to the Q8A format) and block floating point (mantissa of specified bit depth with shared exponent).

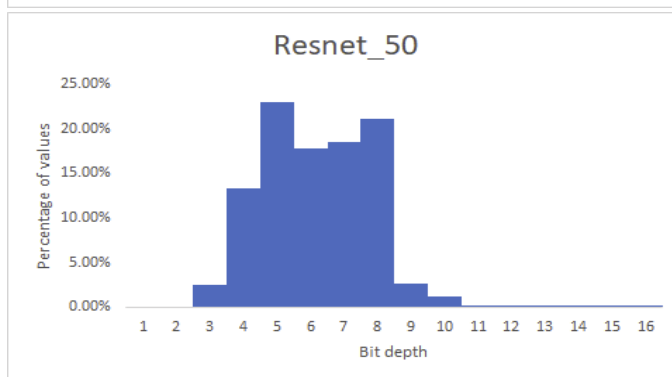
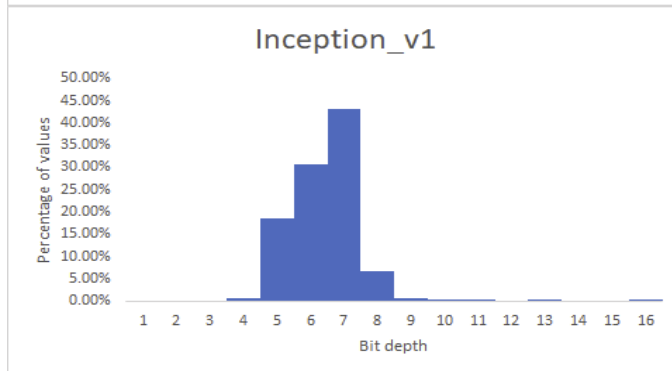
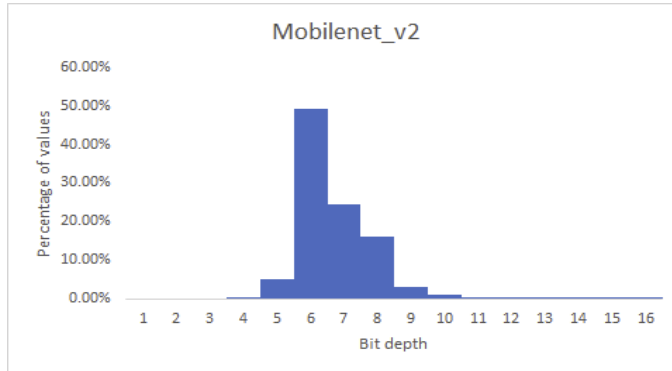
- Floating point scale
- Learned zero offset
- Variable number of bits (not present in the usual Q8A format)

Choice of granularity: for example, shared formats for each tensor or channel.

The following results are for QNA with per-tensor data formats, per-tensor weight bit depths and per-channel weight exponents.











ImageNet Classification

	Original		After Conversion		
	top1	top5	top1	top5	bits/t
Mobilenet v1	70.96	89.80	70.61	89.65	7.24
Mobilenet v2	71.58	90.49	71.46	90.36	6.80
Mobilenet v3	67.55	87.38	67.13	87.24	8.43
Inception v1	69.76	89.54	69.09	89.18	6.34
Resnet v2 (50)	69.73	89.41	67.88	88.19	6.42










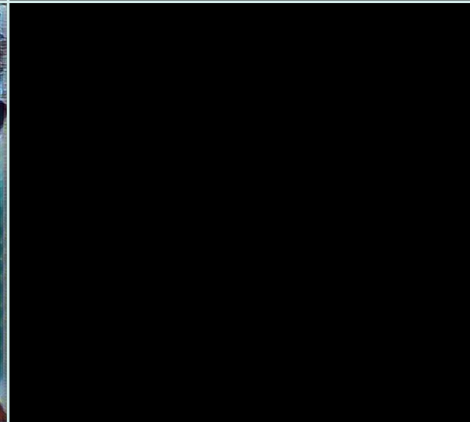
Weight fine tuning is done simultaneously with learning formats.

Weight fine tuning is done simultaneously with learning formats.

	Original	8 bits weighted outlier	8.72 bits (learned formats per tensor)	7.18 bits (learned formats per channel, weight tuning)
	mIOU: 0.669	0.571	0.670	0.668
				
				

Style Transfer

Format selection only – formats were learned based on **one image!** <https://github.com/lengstrom/fast-style-transfer>

Source Image	Original Network	8.75 bits	10 bits weighted outlier
			
			

gives an entirely
blank output at
10 bits using
conventional
method

Conclusion

It is possible to learn suitable number formats, trading off accuracy against compression, using standard deep learning tools.

Combined with distillation, this gives a general approach for optimised conversion of floating point neural networks using small amounts of unlabelled data.

Option to do weight tuning simultaneously (with a larger quantity of unlabelled data).

General cost function that works for a wide variety of tasks.

Dataset security: no need to share labels or large quantities of proprietary data with third parties.

Faithful, sub 8-bit compressed models for ImageNet classification.

Learn more about converting to low precision using distillation

<https://www.imaginationtech.com/blog/low-precision-inference-using-distillation> by Szabolcs Cséfalvay

Key References

Binarized Neural Networks
Hubara et al. NIPS 2016

Distilling the Knowledge in a Neural Network
Hinton et al. NIPS DL & RL Workshop 2015

2021 Embedded Vision Summit

“Deep Neural Networks in Automotive” (Expert Bar)

*Gilberto Rodriguez: Director AI Product Management,
Imagination Technologies*

*Andrew Grant: Senior Director of AI,
Imagination Technologies*

Find out about how DNNs are changing automotive human-machine interfaces, implications for automotive safety and opportunities for accelerating DNNs in automotive applications.