
Image Classification with ResNet

ECE 57000 — Fall 2026

David I. Inouye

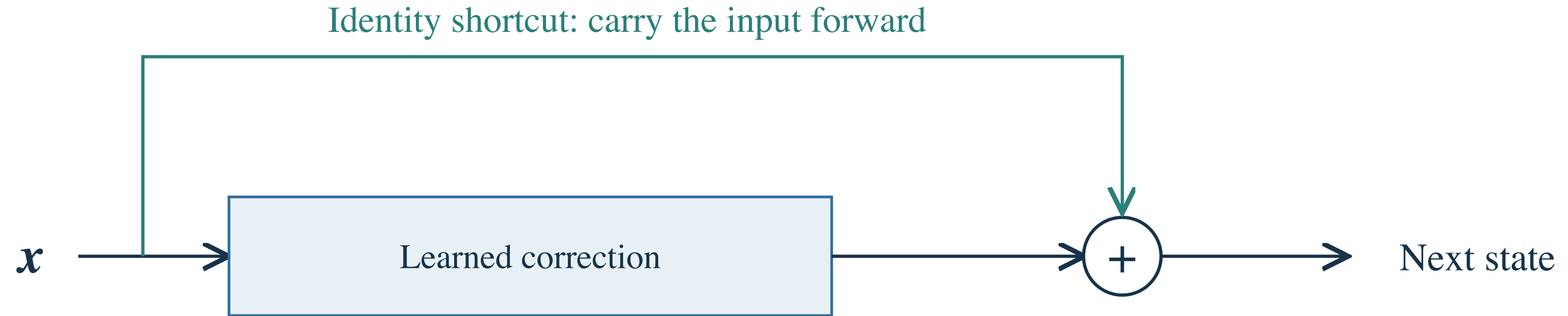
| A realistic classifier combines **updates, normalization, and aggregation**

Arc 5: CNN fundamentals → empirical evidence → **ResNet classification** → U-Net segmentation.

1. **Update:** refine a representation with residual blocks.
2. **Normalize:** control feature scales within the network.
3. **Assemble:** read a complete ResNet-18.

The simple comparison is complete. Now recognize how these components fit into a widely used architecture.

| A residual block learns a change to the representation



If the correction is zero, the representation passes through.

Addition requires matching shapes; use a projection when they differ.

$$\boldsymbol{x}_{\text{next}} = \boldsymbol{x} + f(\boldsymbol{x})$$

| Even a linear residual block starts from a different reference point

Replace the representation

$$x_{\text{next}} = Wx$$

Preserve the input: **learn** $W = I$.

Near $W = 0$, the output is near zero.

Update the representation

$$x_{\text{next}} = (I + W)x$$

Preserve the input: **learn** $W = 0$.

Near $W = 0$, the update is small.

Stacking linear residual blocks still gives a linear map. The change is how we parameterize learning, not the linear model's expressivity.

| The shortcut supplies a **direct gradient contribution**

For the additive block $y = x + f(x)$:

$$\frac{\partial y}{\partial x} = I + \frac{\partial f(x)}{\partial x}.$$

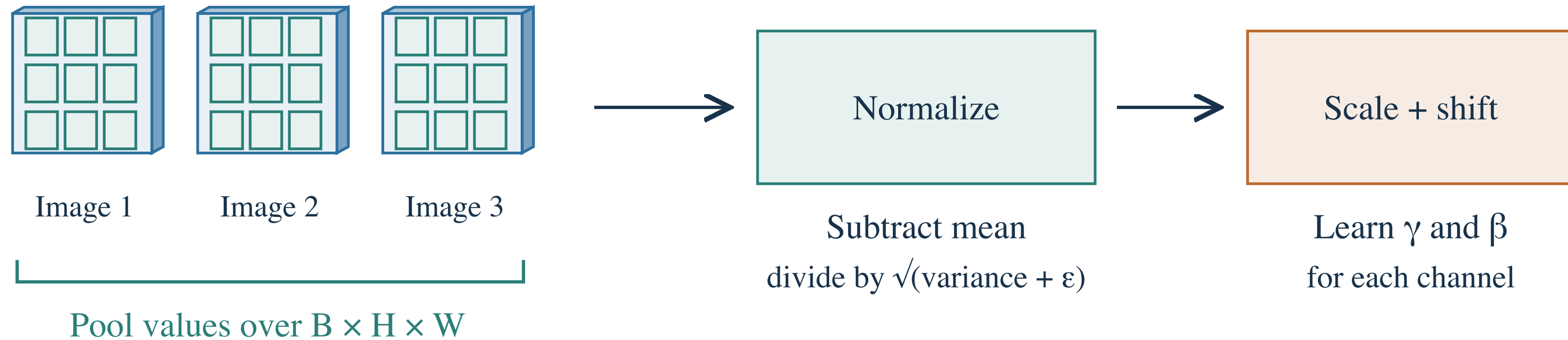
$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} + \left(\frac{\partial f(x)}{\partial x} \right)^T \frac{\partial L}{\partial y}.$$

The gradient has a path that **does not multiply through the learned branch**.

This helps explain deep residual networks; it does not guarantee stable gradients or better generalization.

| Batch normalization uses **one set of statistics per channel**

One channel across a batch



Training: batch statistics • Evaluation: stored running statistics

$$y_{bchw} = \gamma_c \frac{x_{bchw} - \mu_c}{\sqrt{v_c + \epsilon}} + \beta_c$$

| Normalization layers differ in **which values share statistics**

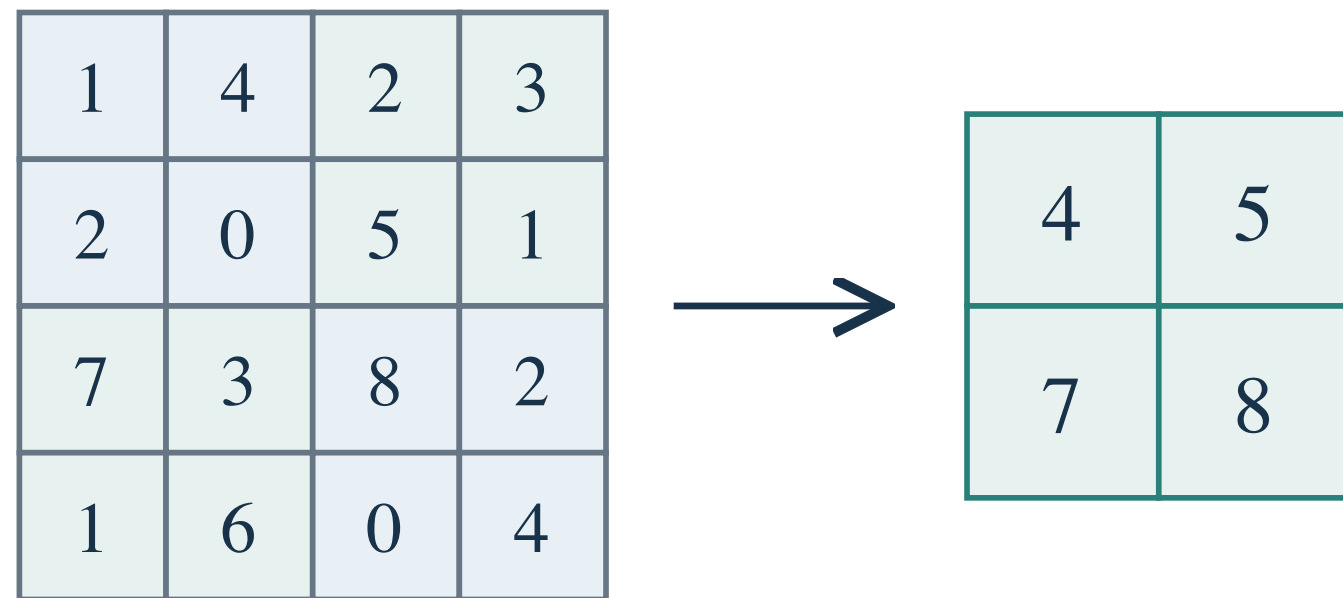
Layer	Values grouped for each normalization
BatchNorm	One channel across examples and spatial positions
InstanceNorm	One channel's spatial positions within one example
GroupNorm	A group of channels and spatial positions within one example
LayerNorm	Chosen trailing feature dimensions within one example

ResNet-18 uses BatchNorm. The U-Net example uses InstanceNorm.

The other three need no cross-example statistics. LayerNorm will return when we discuss transformers.

| Pooling either reduces a map or summarizes each channel

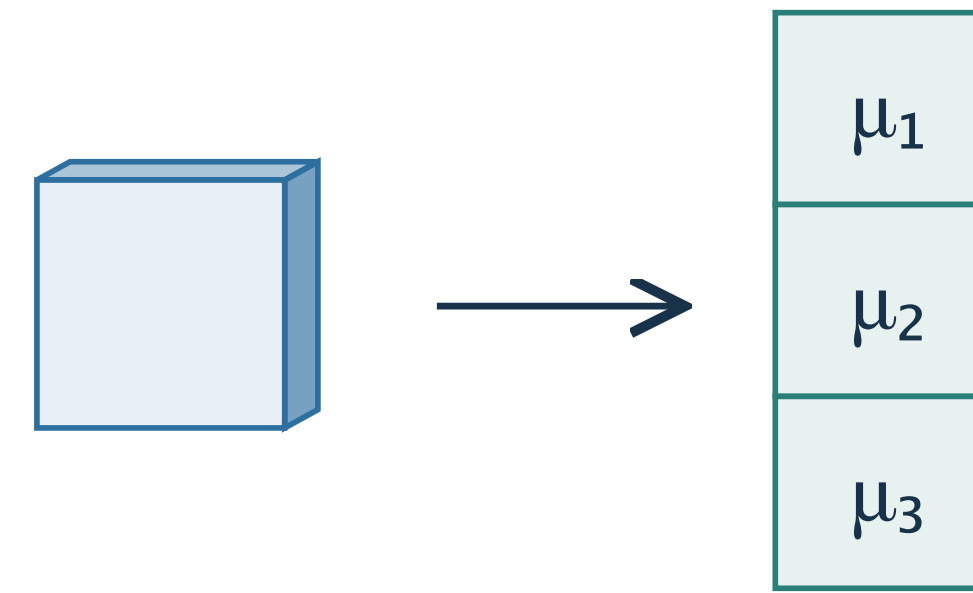
Local max pooling



Keep the largest value in each 2×2 patch

$4 \times 4 \rightarrow 2 \times 2$, separately per channel

Global average pooling



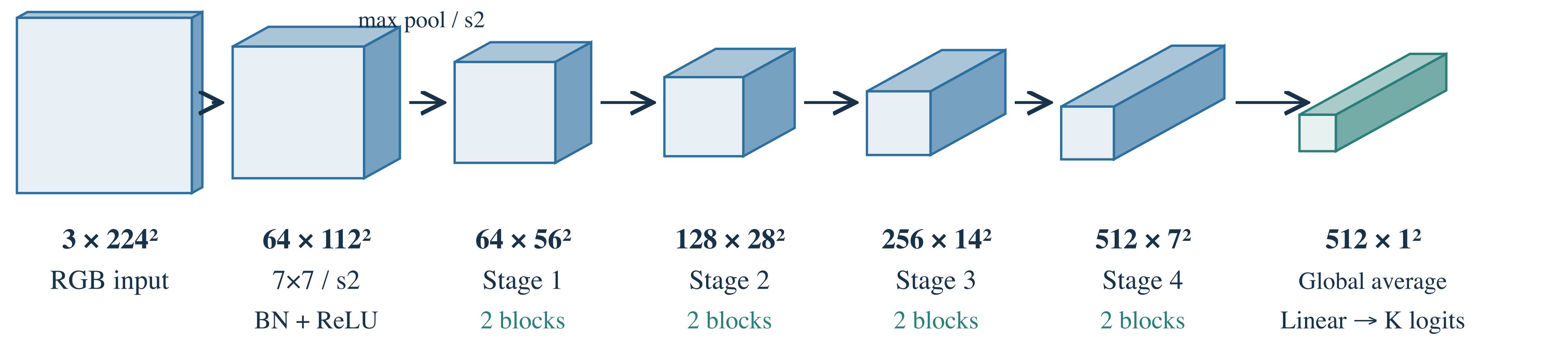
Average each channel over all locations

$C \times H \times W \rightarrow C$ values

ResNet uses both: local pooling near the input; global averaging before the classification head.

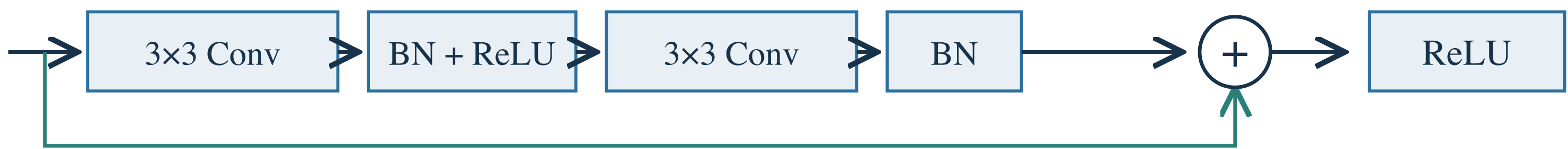
| ResNet-18 uses smaller spatial maps and more channels

ResNet-18 • RGB image classification



Front face: height $\times$ width • Box depth: channels • Sizes compressed for readability

Inside each basic block



The shortcut preserves shape or projects to match it.

At stage changes:

shortcut: $1 \times 1 / s2$
Conv + BN

Original illustration of torchvision ResNet-18; dimensions omit the batch axis.