Aprendizaje Automático Profundo (Deep Learning)

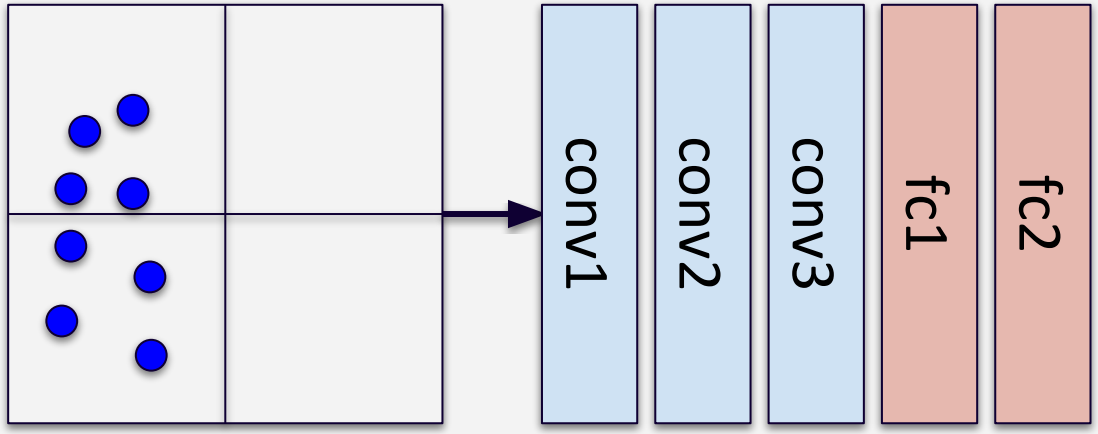
Dr. Facundo Quiroga - Dr. Franco Ronchetti

Capa Batch normalization
(normalización por lotes)

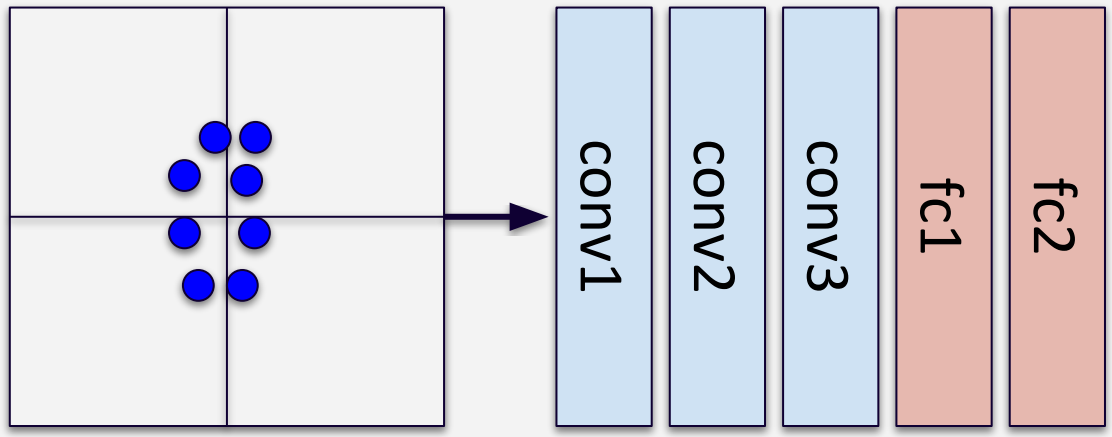
Normalización de la entrada

- ¿Por qué normalizamos los datos?
  - Mejora el aprendizaje
  - Mejora el accuracy
  - Estandariza las magnitudes
    - Comparables a través de modelos/datasets

Sin normalizar



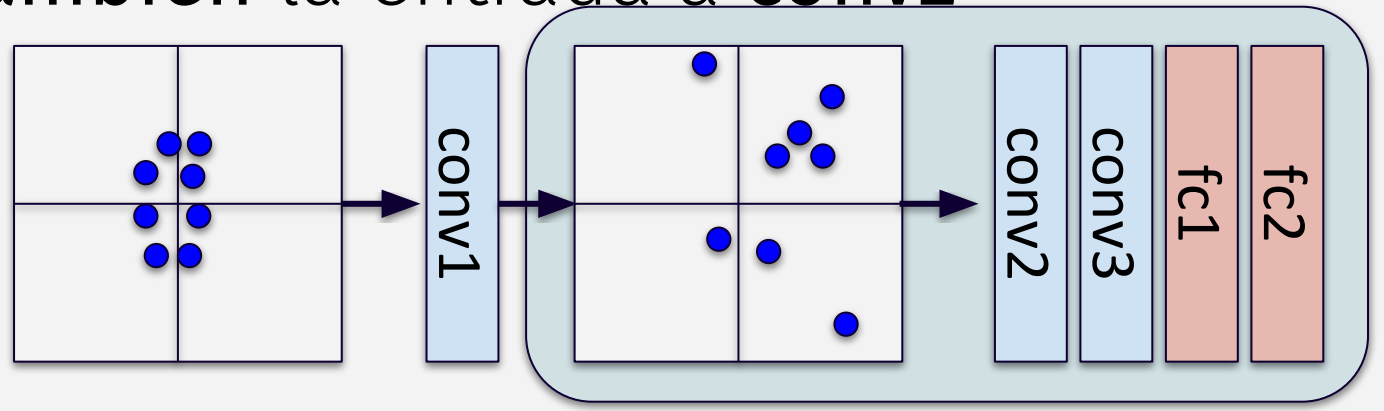
Normalización Z



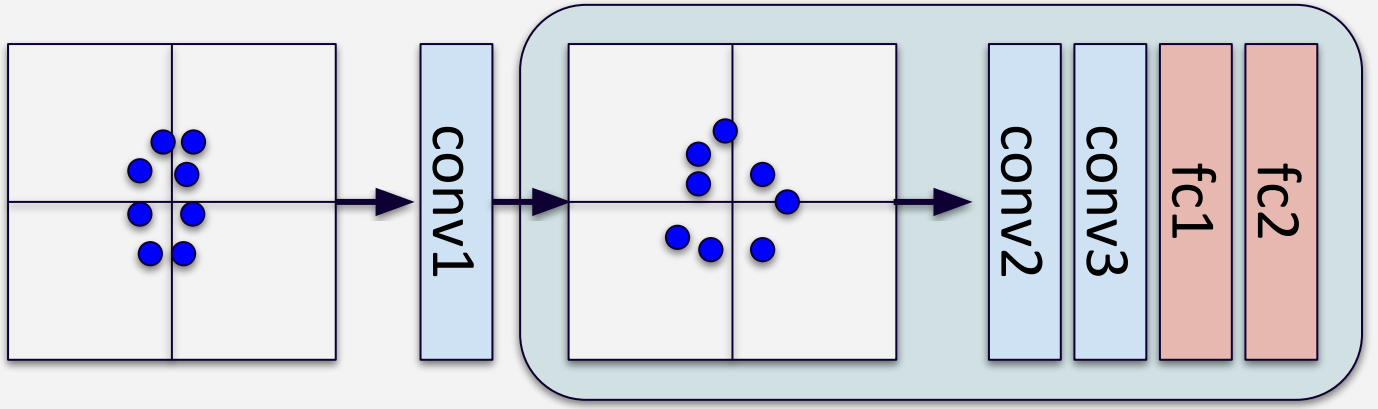
Capa Batch Normalization (BN)

- ¿Qué pasa entre capas (conv1 y conv2)?
- Estructura recursiva de las redes
  - Una red sin su primer capa también es una red
    - Idea: Normalicemos **también** la entrada a **conv2**

- Situación actual

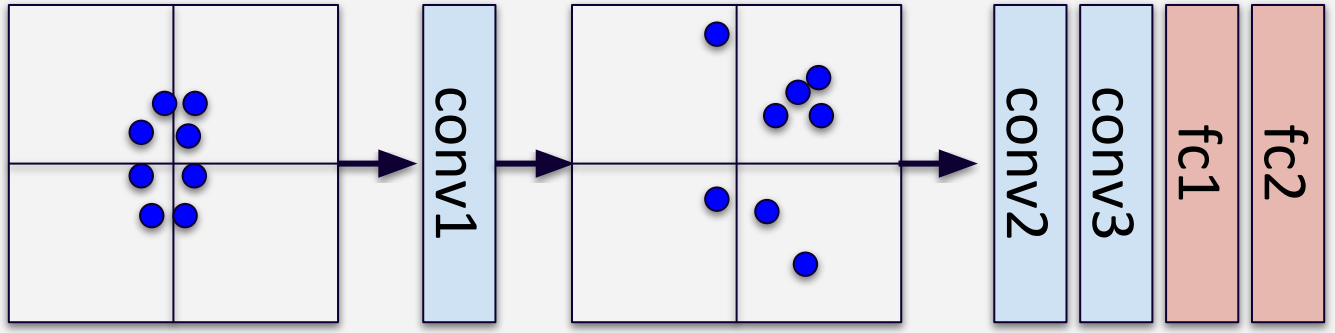


- Situación ideal

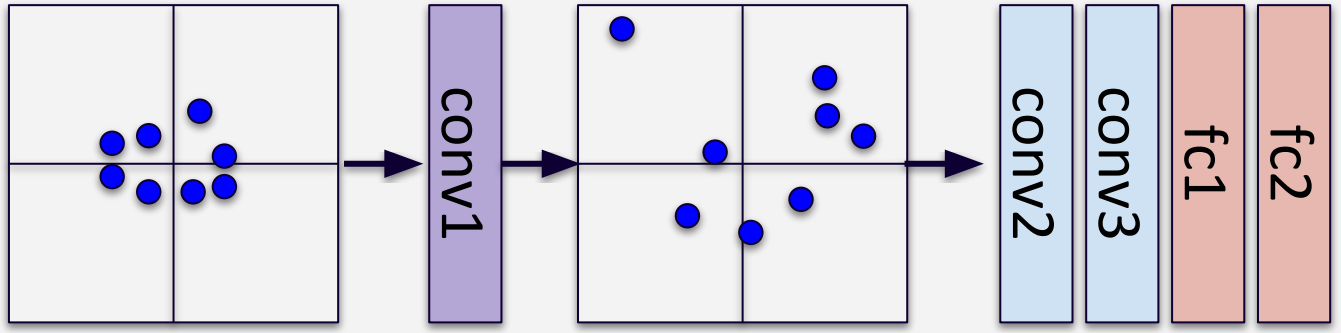


Capa Batch Normalization (BN)

- Problema durante el entrenamiento
  - La salida de conv1 cambia luego de cada batch
    - Cambia el batch y parámetros!
- Batch 1: conv1 tiene pesos w
  - Se actualizan
  - w := w - dw * alpha



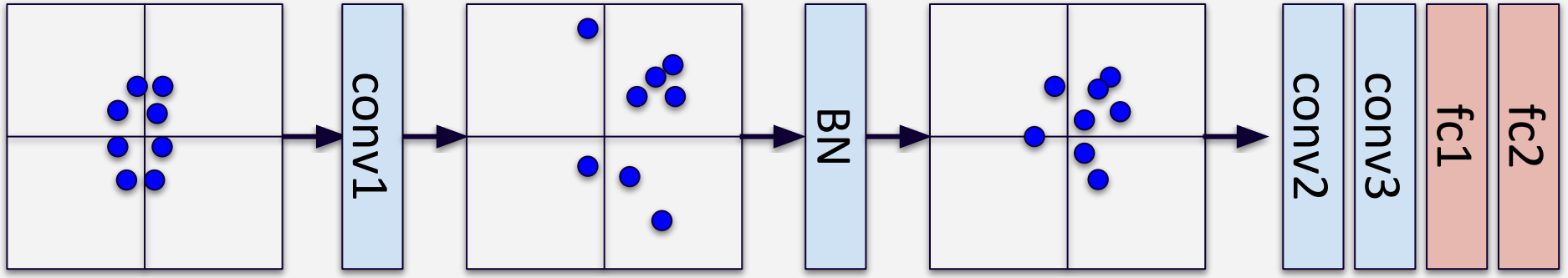
- Batch 2: conv1 tiene pesos w
  - Se actualizan
  - w := w - dw * alpha
  - (mismo para batch N)



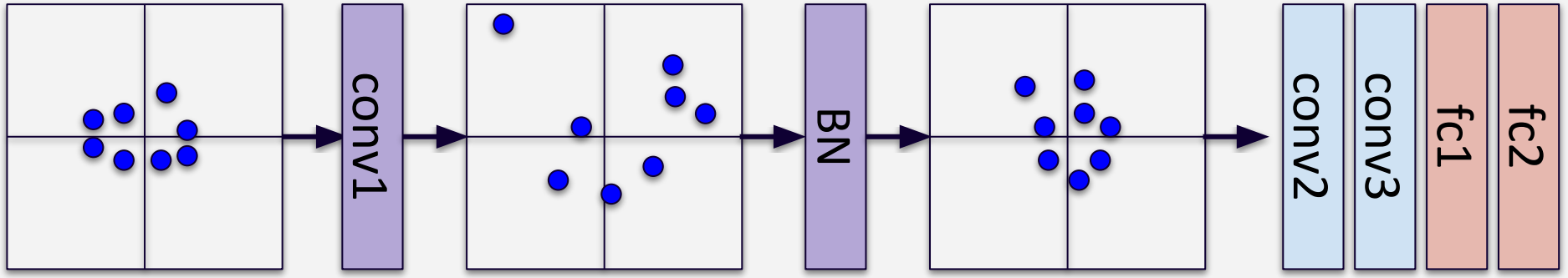
Capa Batch Normalization (BN)

- Solución
  - Normalización Z en cada batch
    - Estimar parámetros (μ y σ) para cada batch

- Batch 1:
  - BN calcula μ y σ
  - Normaliza



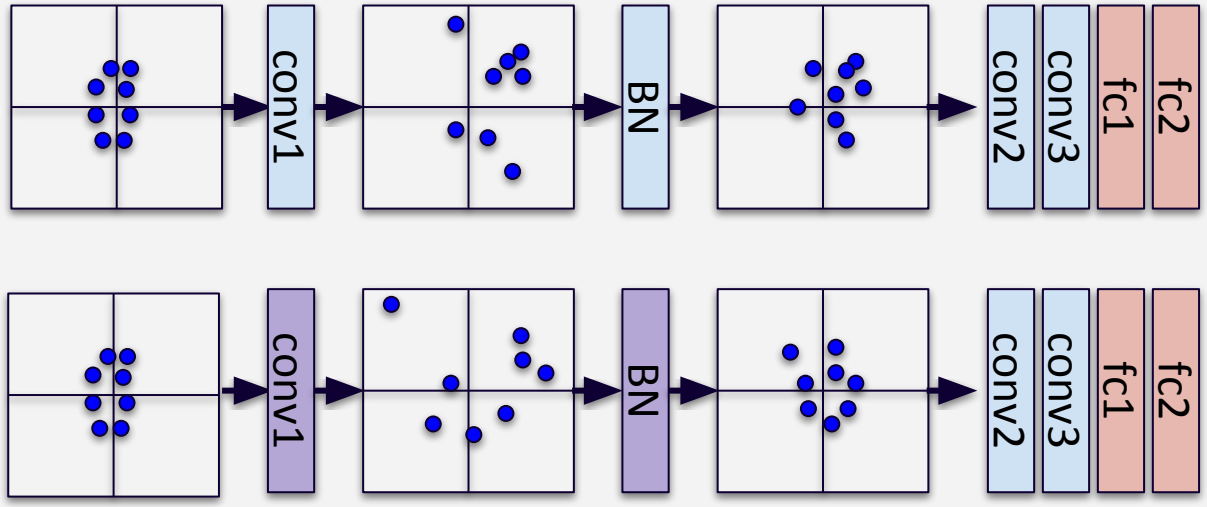
- Batch 2:
  - BN calcula μ y σ
  - Normaliza



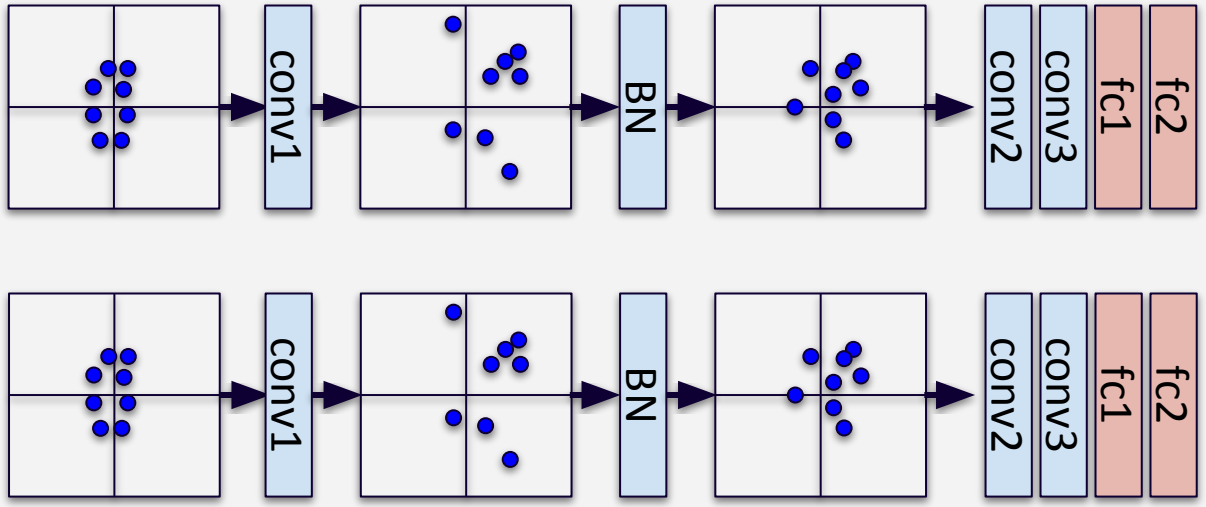
Resumen

- Capa Batch Normalization (BN o batchnorm)
  - Tiene parámetros μ y σ para normalizar entre capas
  - Acelera el entrenamiento

- Durante el entrenamiento μ y σ se actualizan



- En ejecución μ y σ quedan fijos



Capa Batch Normalization (BN) en Keras

- Muy fácil de usar (notebook)

```
model = Sequential()
model.add(Conv2D(conv_filters,activation="relu", ...))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(dense_filters,activation="relu"))
model.add(BatchNormalization())
model.add(Dense(classes,activation="softmax"))
```

- Alternativa: BatchNormalization ANTES de ReLU (puede funcionar mejor)

```
model.add(Dense(dense_filters,activation=None))
model.add(BatchNormalization())
model.add(Activation("relu")) #ReLU “desactiva”. BN no la deja
```

Detalles avanzados (paper)

- El cálculo de **μ** y **σ** no es parte de la optimización (descenso de gradiente)
- BN agrega dos parámetros **γ** y **β** que sí se optimizan
  - Restauran poder de expresividad perdido por la normalización
- Posiblemente, mantiene los **autovalores** de los tensores intermedios en un régimen estable
  - Los valores intermedios no son todos 0, ni son muy grandes
- Tiene muy poco coste computacional
- Como el cálculo de **μ** y **σ** es por batch, agregan un poco de ruido
  - Sirve como leve regularización del modelo
- No se entiende completamente todavía
  - Pero se usa porque entrena más rápido