Aprendizaje profundo

Regularización y entrenamiento de redes profundas

Gibran Fuentes-Pineda

Noviembre 2020

Preprocesamiento de datos

Preprocesamiento

- Re-escalado

$$\mathbf{x}' = \frac{\mathbf{x} - min(\mathbf{x})}{max(\mathbf{x}) - min(\mathbf{x})}$$

- Estandarización

$$\mathbf{x}' = \frac{\mathbf{x} - \bar{x}}{\sigma}$$

- Magnitud unitaria

$$\mathbf{x}' = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

Preprocesamiento: imágenes

- Radio uniforme

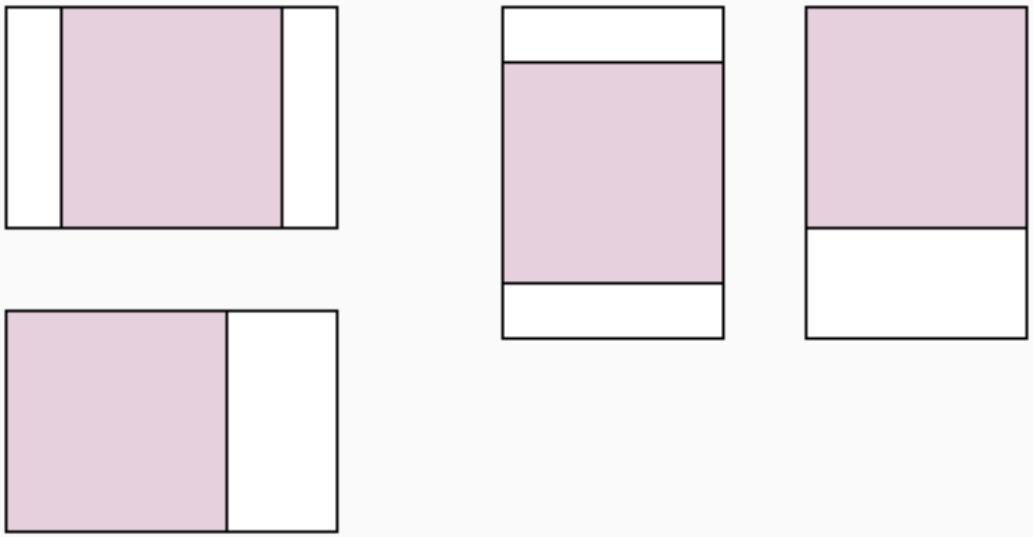


Imagen tomada de Nikhil B. Image Data Pre-Processing for Neural Networks, 2017.

Preprocesamiento: imágenes

- Radio uniforme
- Escalado de valores de pixeles

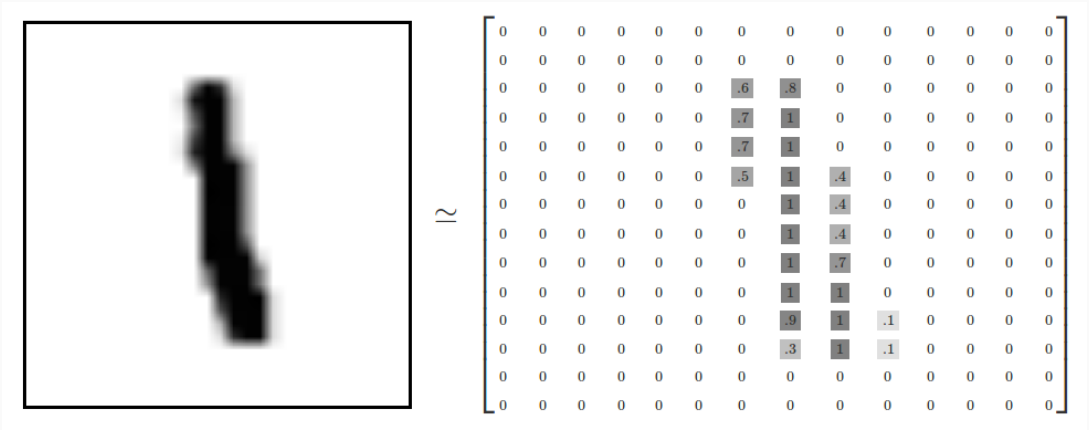


Imagen tomada de presentación de Yubei Chen. Part I: Manifold Learning, 2018.

Preprocesamiento: audio

- Filtrado

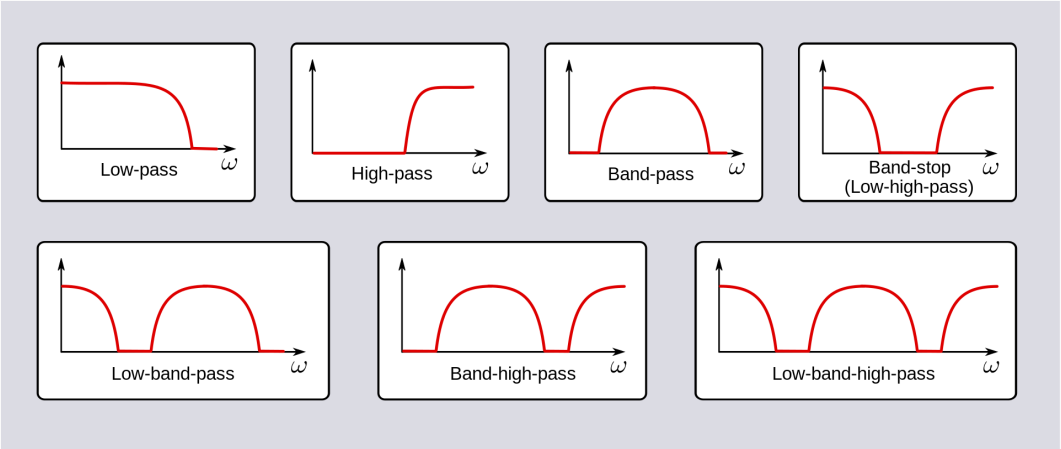


Imagen del usuario SpinningSpark de Wikipedia (entrada Filter (signal processing)). CC BY-SA 3.0

Preprocesamiento: audio

- Filtrado
- Espectograma

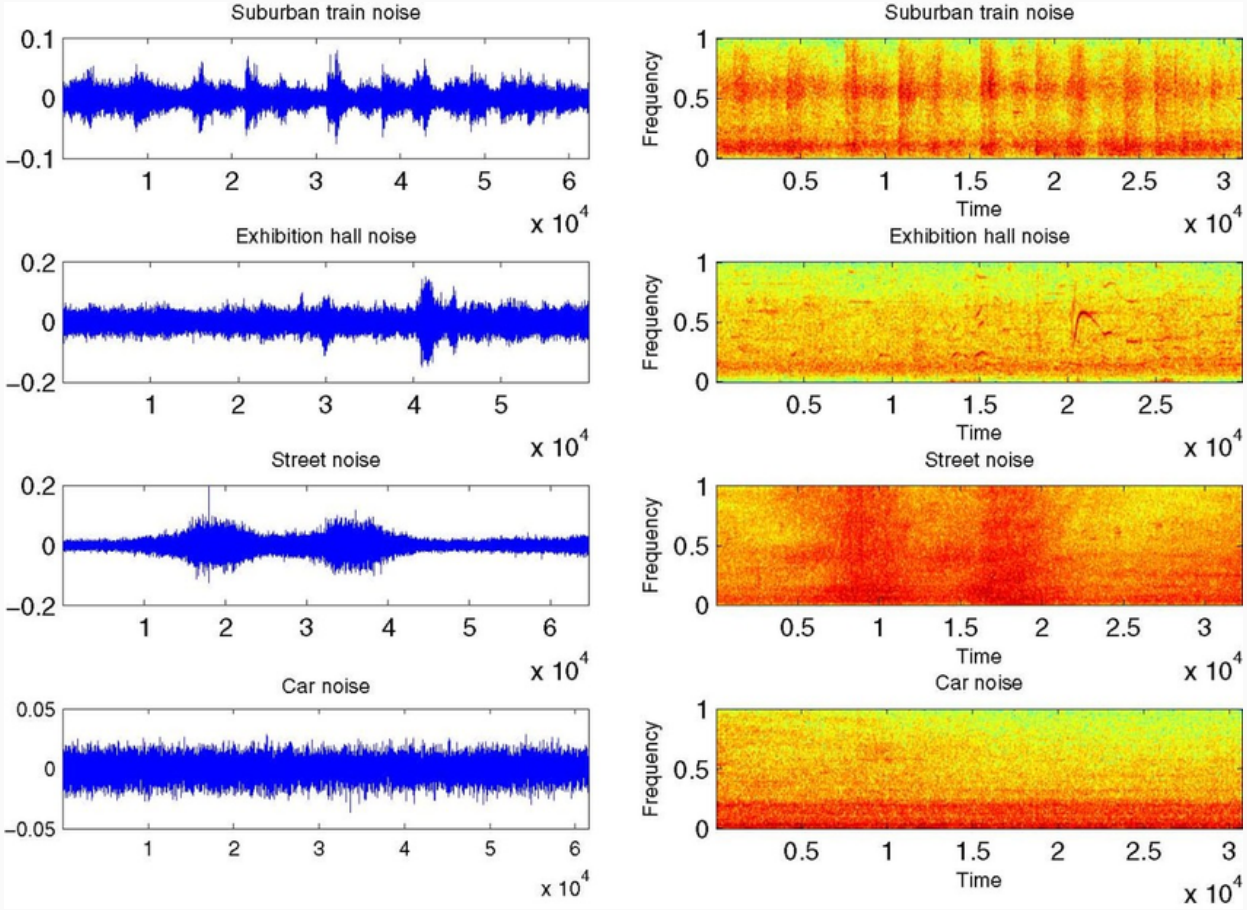


Imagen tomada de Zouhir and Ouni. A bio-inspired feature extraction for robust speech recognition, 2014.

Preprocesamiento: texto como bolsas

- Bolsas de palabras
  - Stopwords
  - Stemming
  - Quitar caracteres

Preprocesamiento: representaciones distribuidas

- Encajes de palabra

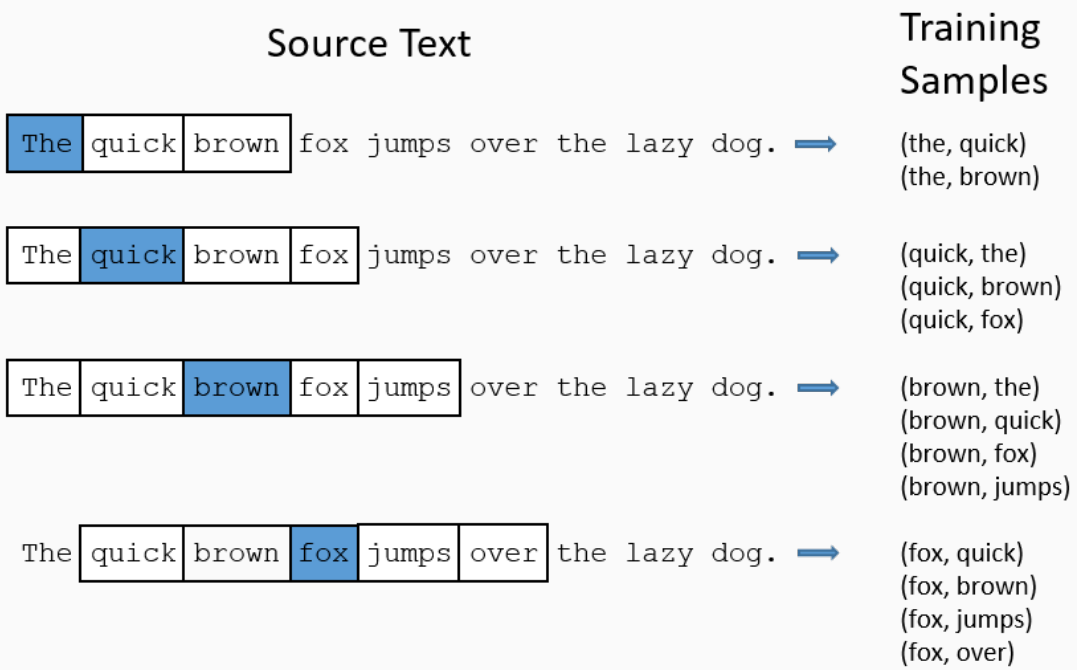


Imagen tomada de McCormick. Word2Vec Tutorial – The Skip-Gram Model, 2016.

Preprocesamiento: palabras como vectores densos

- Encajes de palabra
- Word2Vec

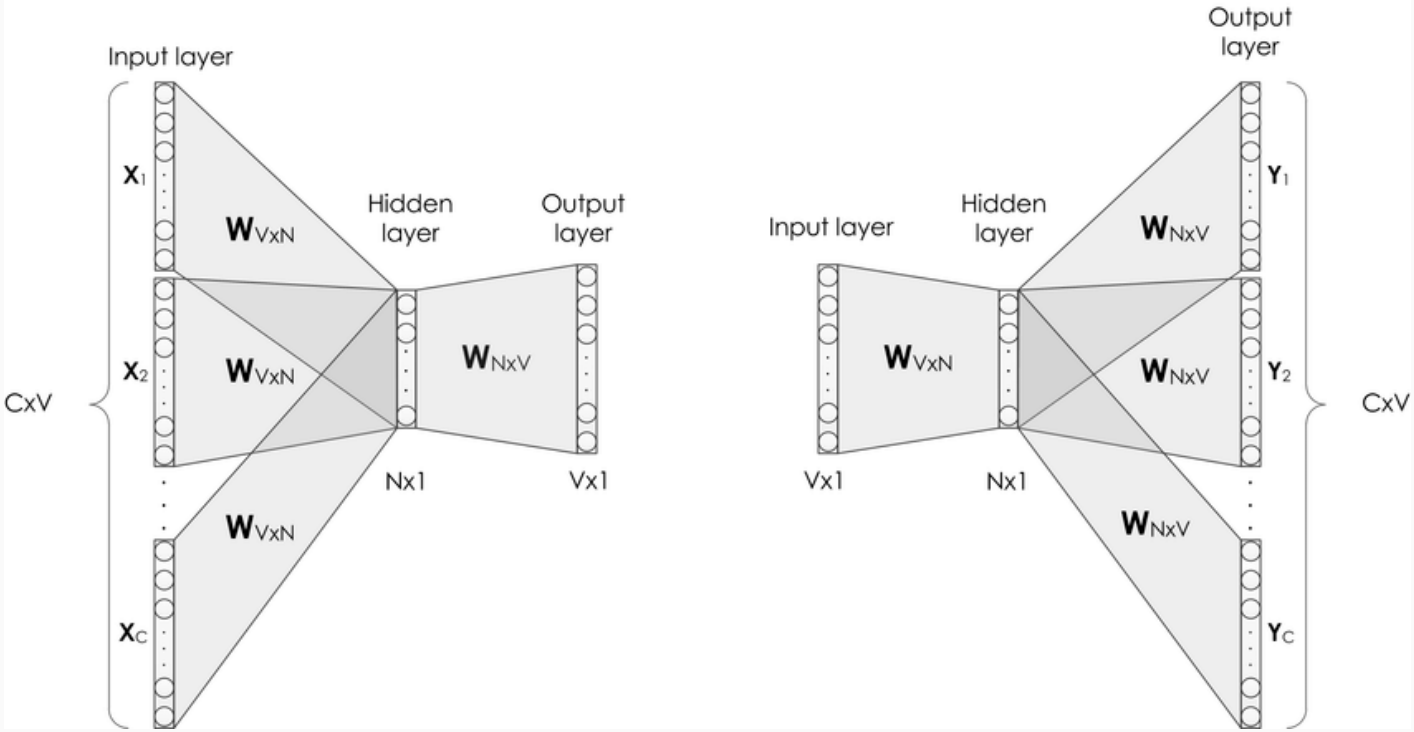


Imagen tomada de Tutubalina y Nikolenko. Demographic Prediction Based on User Reviews about Medications, 2017.

Preprocesamiento: video

- Preprocesamiento por marco
- Muestreo de marcos (por ej., uniforme o por movimiento)
- Flujo de movimiento

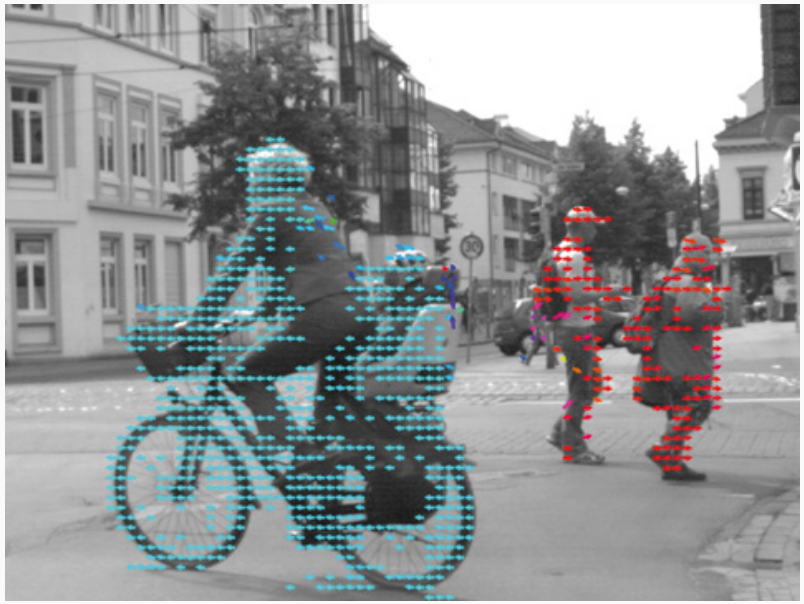


Imagen tomada de `https://www.commonlounge.com/discussion/1c2eaa85265f47a3a0a8ff1ac5fbce51`

Acrecentamiento de datos

Acrecentamiento de imágenes: transformaciones

- Traslación

$$\begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

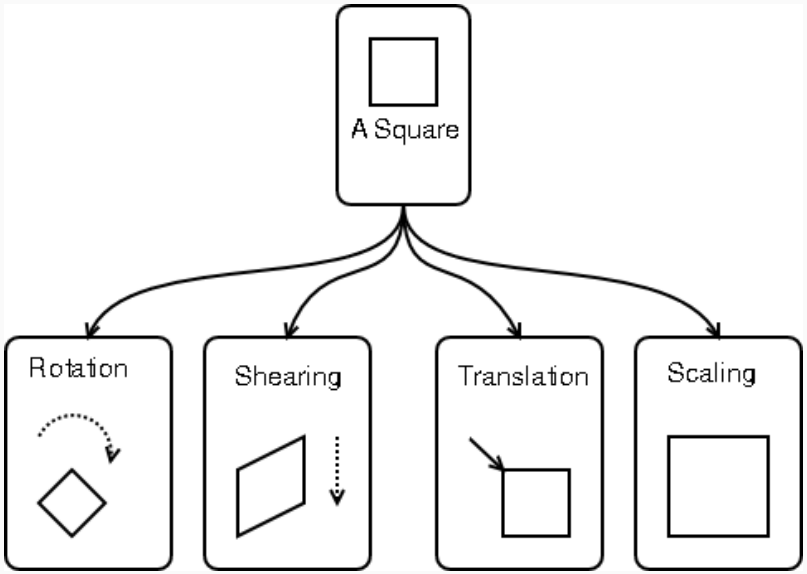


Imagen tomada de `https://people.gnome.org/~mathieu/libart/libart-affine-transformation-matrices.html`

Acrecentamiento de imágenes: transformaciones

- Traslación
- Rescalado

$$\begin{bmatrix} v_x & 0 & 0 \\ 0 & v_y & 0 \\ 0 & 0 & v_z \end{bmatrix}$$

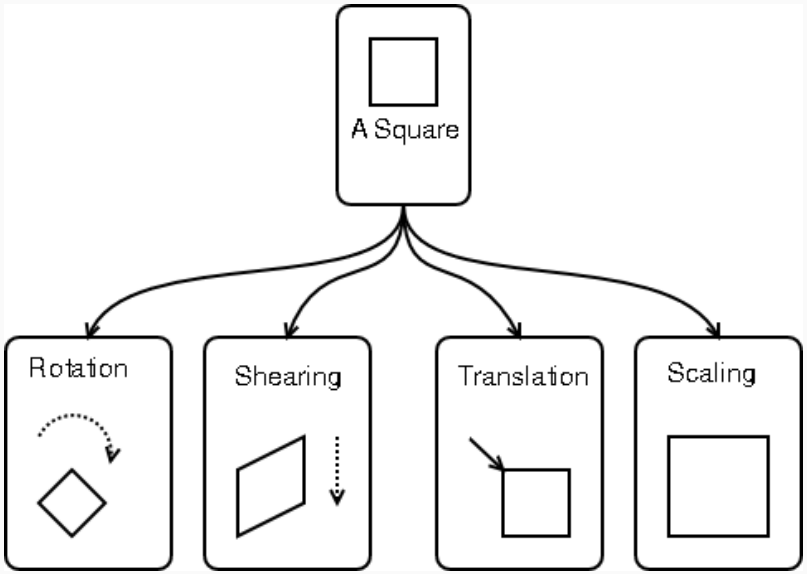


Imagen tomada de `https://people.gnome.org/~mathieu/libart/libart-affine-transformation-matrices.html`

Acrecentamiento de imágenes: transformaciones

- Traslación
- Rescalado
- Giro

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

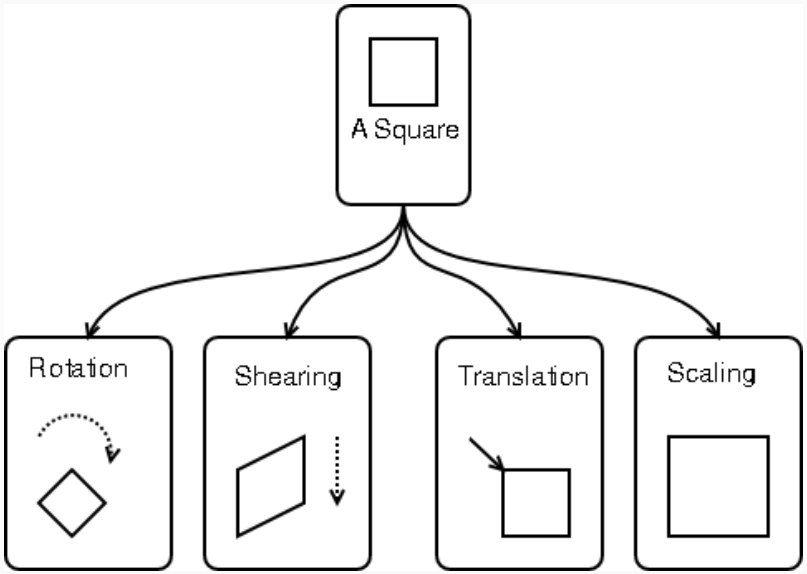


Imagen tomada de `https://people.gnome.org/~mathieu/libart/libart-affine-transformation-matrices.html`

Acrecentamiento de imágenes: transformaciones

- Traslación
- Rescalado
- Giro
- Rotación

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

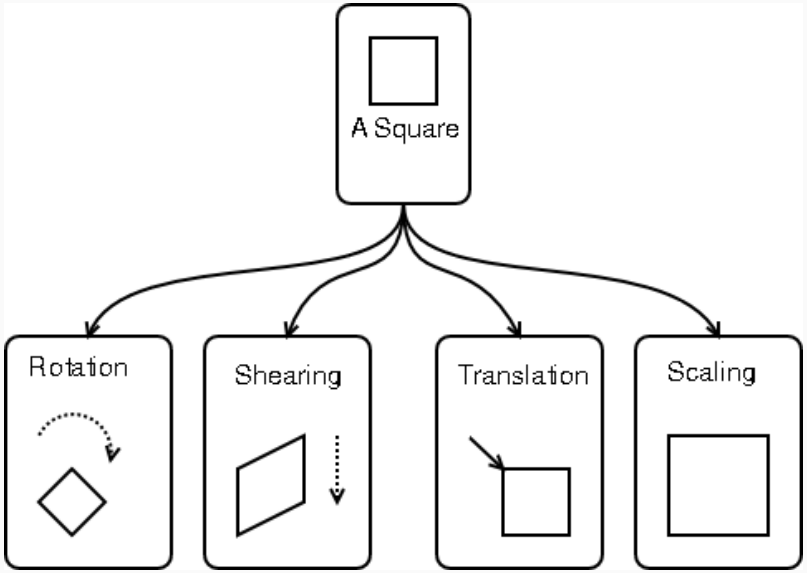


Imagen tomada de `https://people.gnome.org/~mathieu/libart/libart-affine-transformation-matrices.html`

Acrecentamiento de imágenes: transformaciones

- Traslación
- Rescalado
- Giro
- Rotación
- Deformación de corte

$$\begin{bmatrix} 1 & c_x = 0.5 & 0 \\ c_y = 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

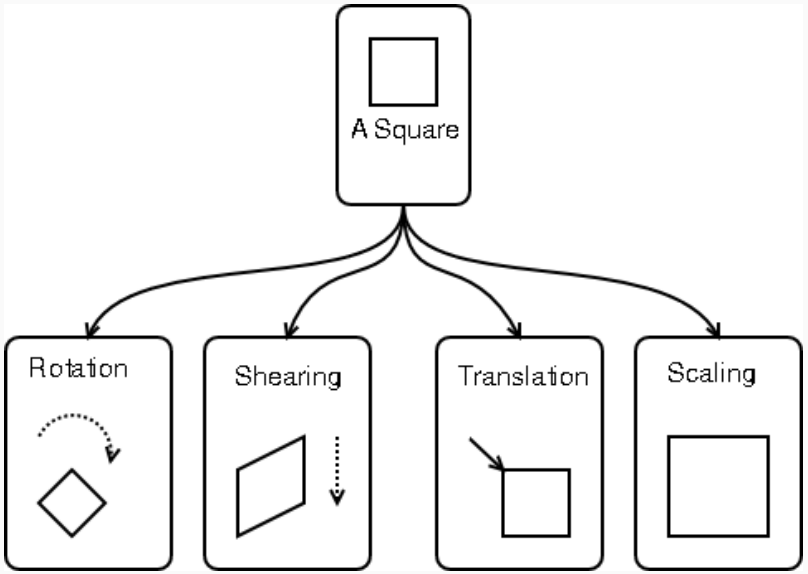


Imagen tomada de https://people.gnome.org/~mathieu/libart/libart-affine-transformation-matrices.html

Acrecentamiento de imágenes: transformaciones

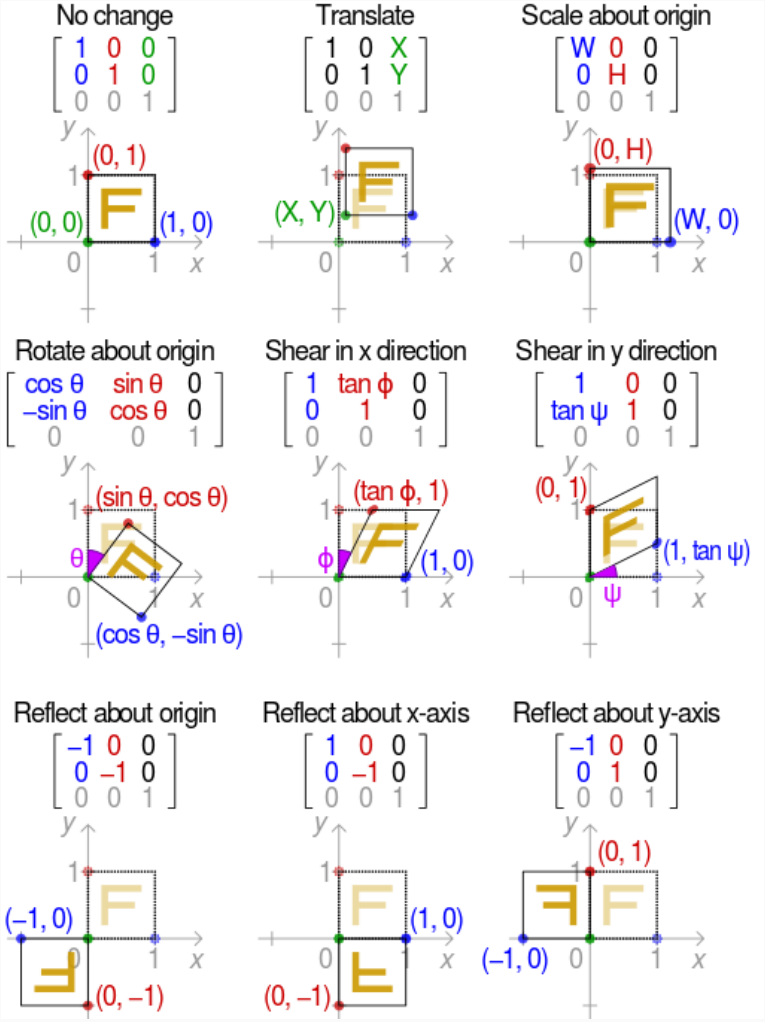


Imagen del usuario de Wikipedia Cmglee (entrada Affine transformation). CC BY-SA 3.0

Acrecentamiento de imágenes: ejemplos

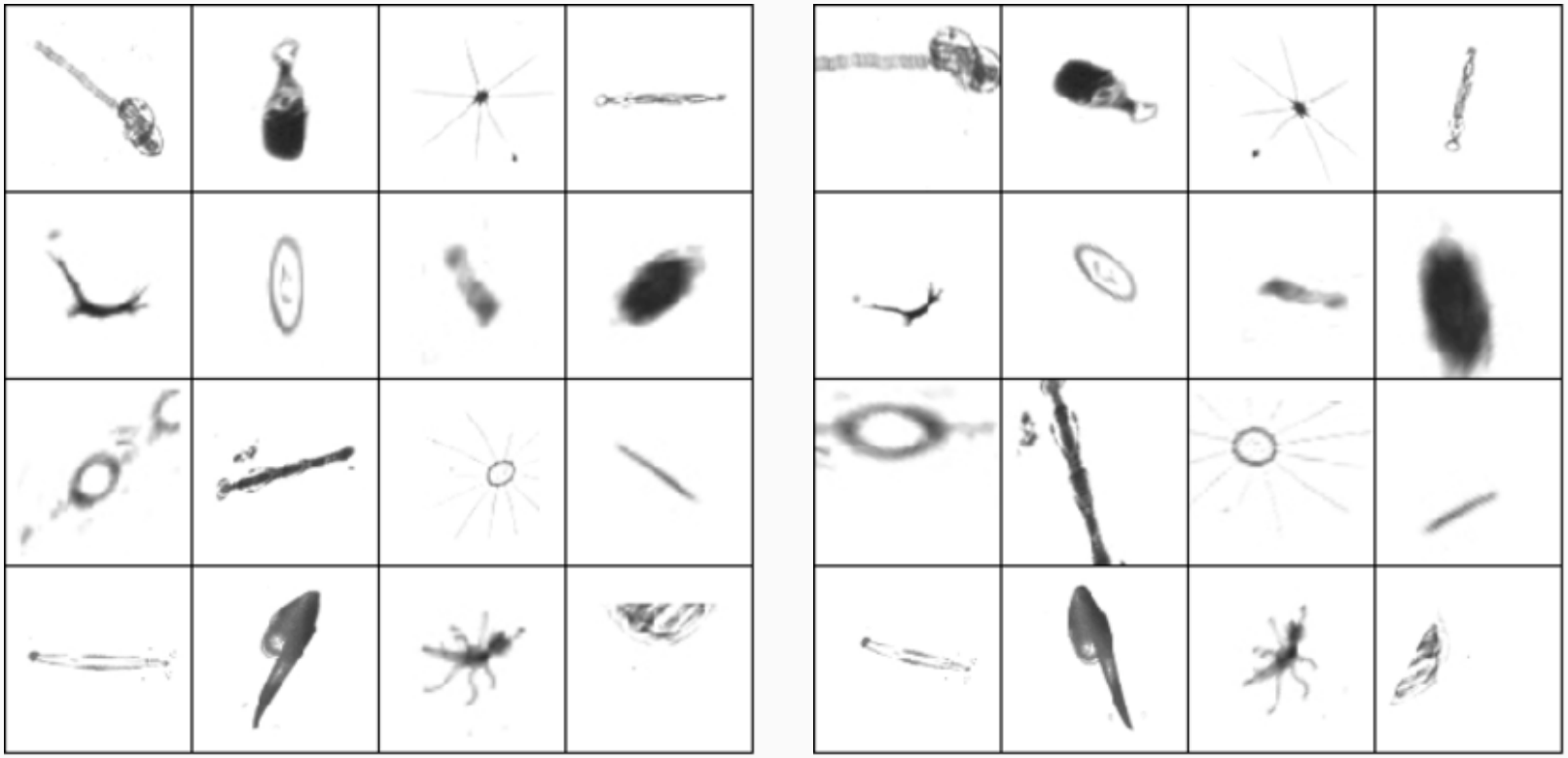


Imagen tomada de Dieleman. Classifying plankton with deep neural networks, 2015.

Acrecentamiento de audio

- Ruido aditivo

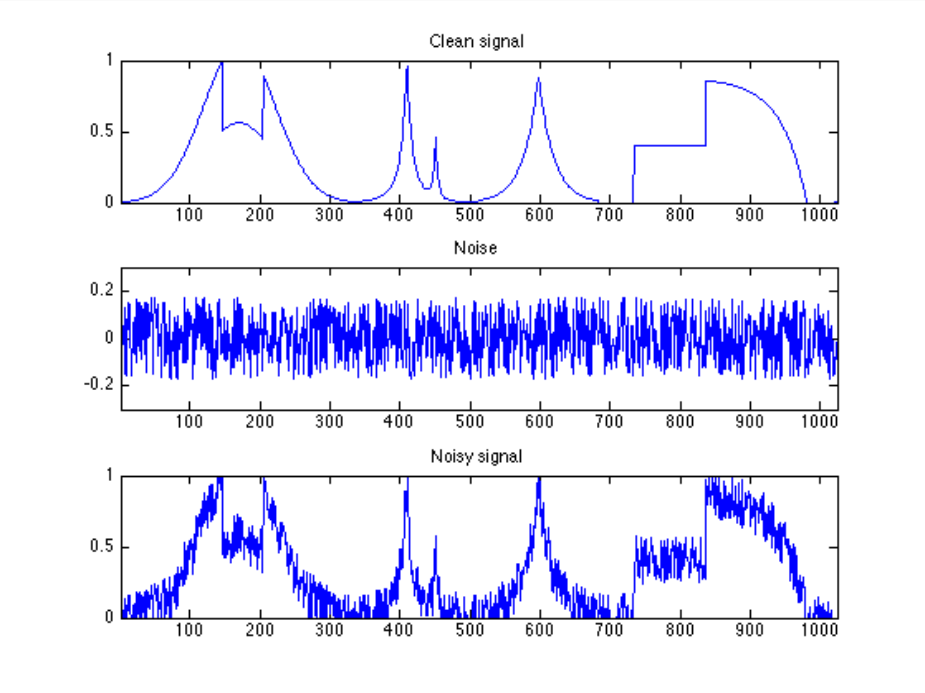


Imagen tomada de Peyre. Signal and Image Noise Models, 2008

Acrecentamiento de audio

- Ruido aditivo
- Enmascaramiento del espectograma

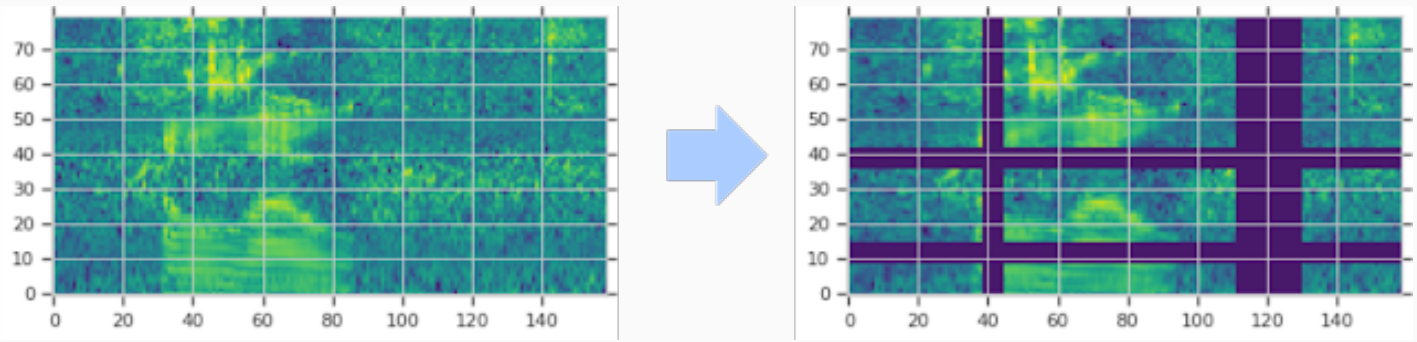


Imagen tomada de `https://ai.googleblog.com/2019/04/specaugment-new-data-augmentation.html`

Acrecentamiento en texto

- Símbolos
  - Insertar/cambiar/quitar símbolos aleatoriamente
  - Simular errores de teclado
  - Simular errores de OCR
- Palabra
  - Cambiar/quitar palabras aleatoriamente o con algún modelo de lenguaje o bolsa de palabras
  - Cambiar palabras por sinónimos
  - Cambiar palabras de acuerdo a errores de escritura

Aprendizaje por transferencia

Esquema general

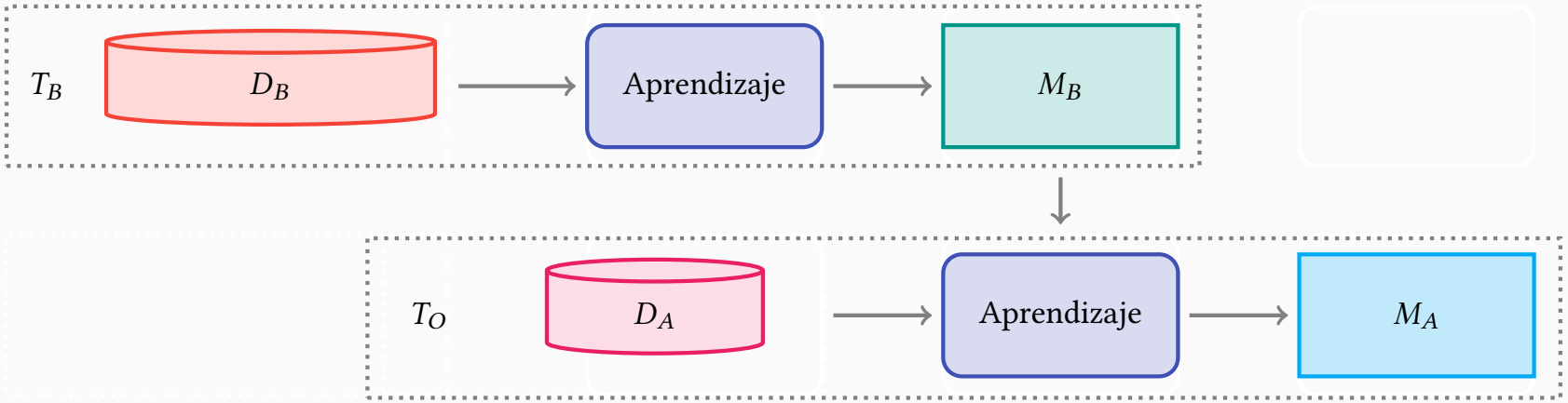


Imagen cortesía de Berenice Montalvo

Desempeño en distintas tareas

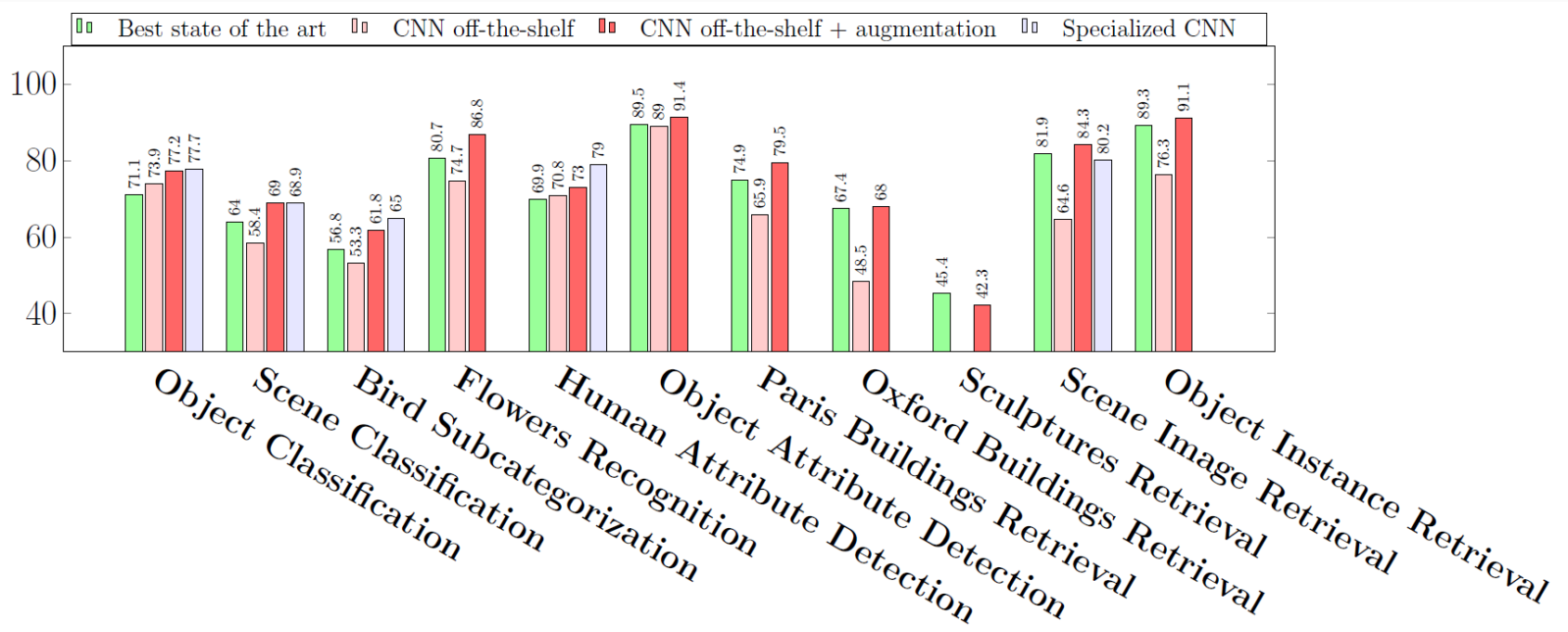


Imagen tomada de Razavian et al. CNN Features off-the-shelf: an Astounding Baseline for Recognition, 2014

Desempeño en tareas con distintas características (1)

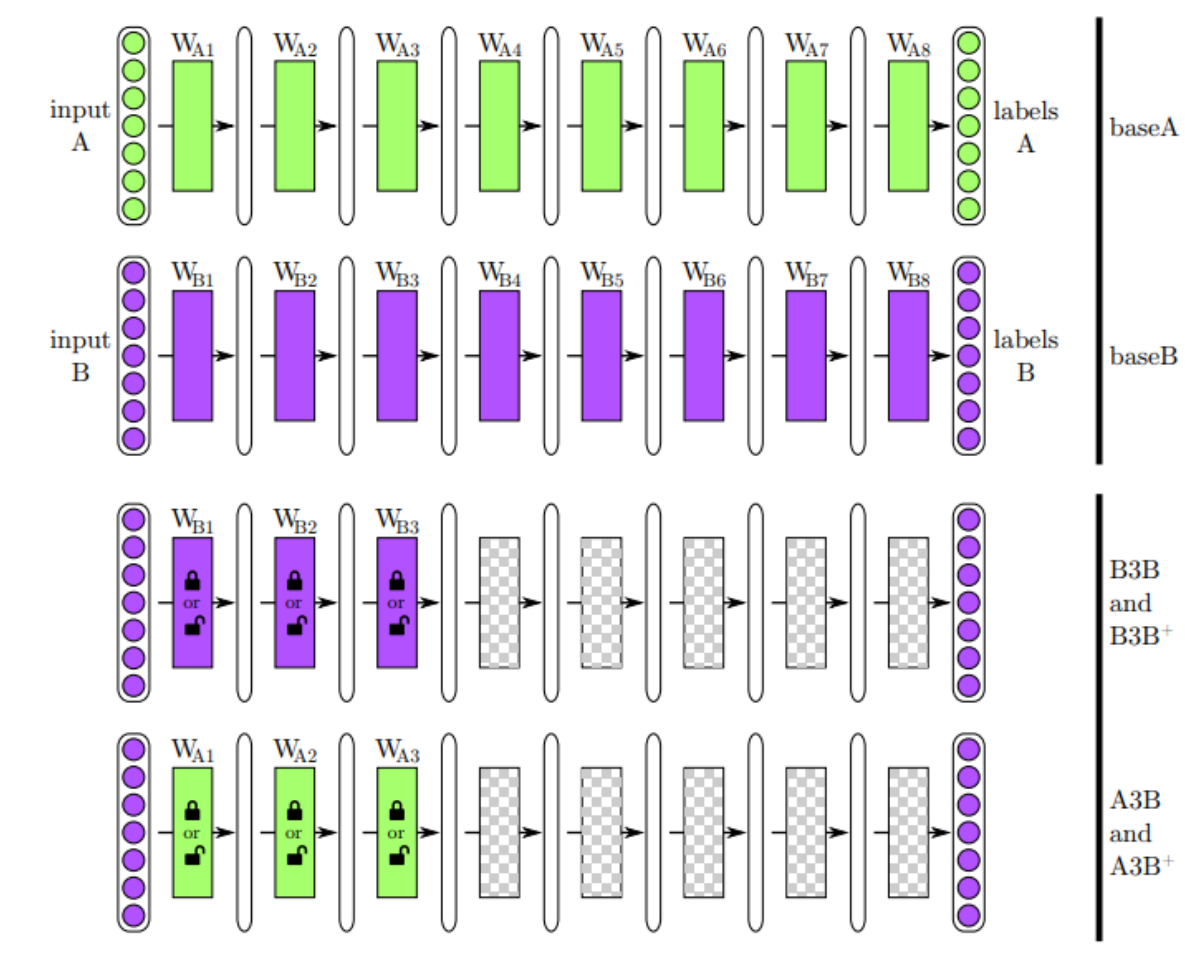


Imagen tomada de Yosinski et al. How transferable are features in deep neural networks?, 2014

Desempeño en tareas con distintas características (2)

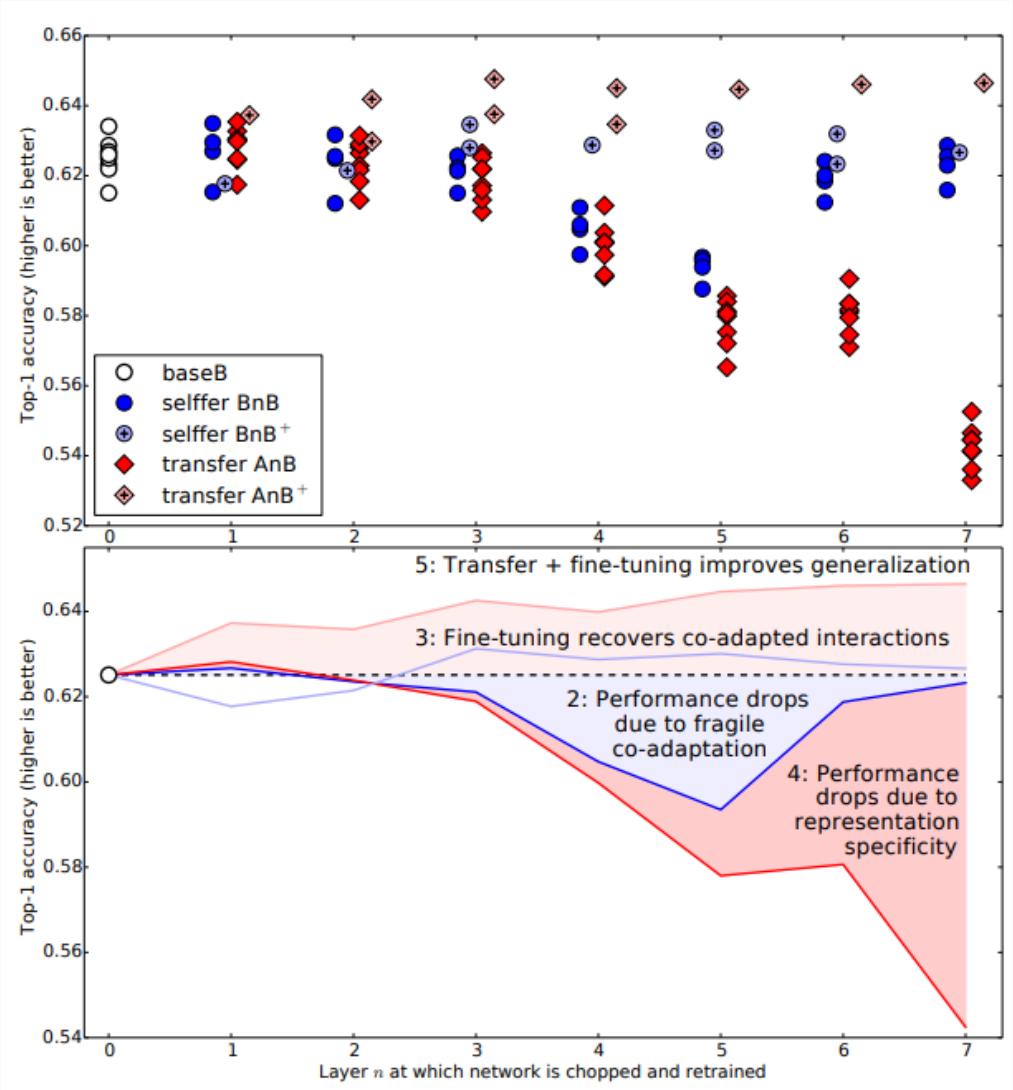


Imagen tomada de Yosinski et al. How transferable are features in deep neural networks?, 2014

Sobreajuste y cómo atacarlo

Regularización

- Estrategias para reducir error de generalización:
  - Penalización de función de error (o función de pérdida)
  - Adición de ruido a entradas, salidas y/o parámetros
  - Ensambles
  - Paro temprano
  - Aprendizaje de múltiples tareas
  - Dropout
  - Normalización por lotes

Penalizando pesos y sesgos con norma ℓ_1 y ℓ_2

- Norma ℓ_1

$$\tilde{E}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\{y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\} + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_1$$

- Norma ℓ_2

$$\tilde{E}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\{y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\} + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$$

Paro temprano

- Para entrenamiento si pérdida o métrica de validación no aumenta después de varios pasos
- Usualmente se elige el modelo con mejor desempeño en el conjunto de validación

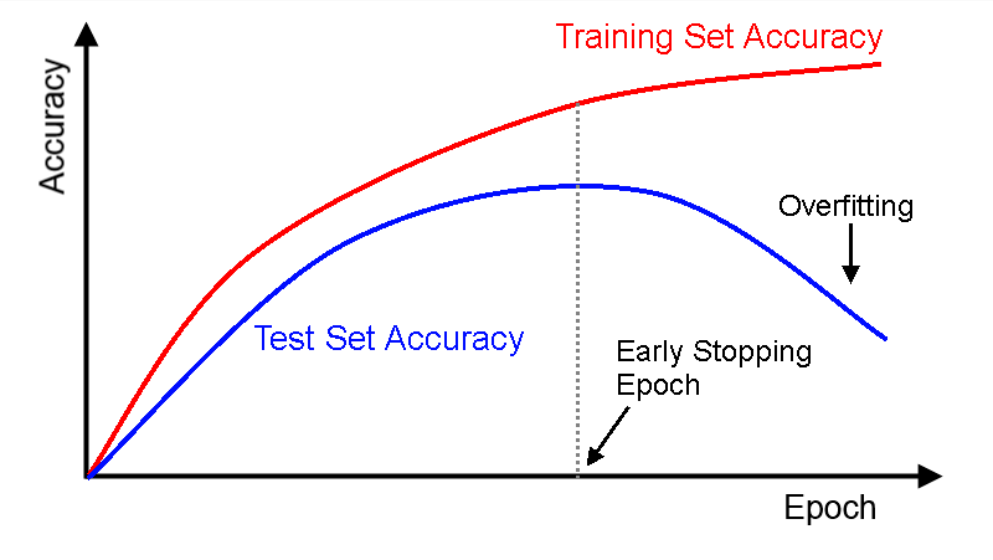


Imagen tomada de `https://deeplearning4j.org/earlystopping`

Aprendizaje de múltiples tareas

- Tener una representación genérica compartida entre 2 tareas relacionadas

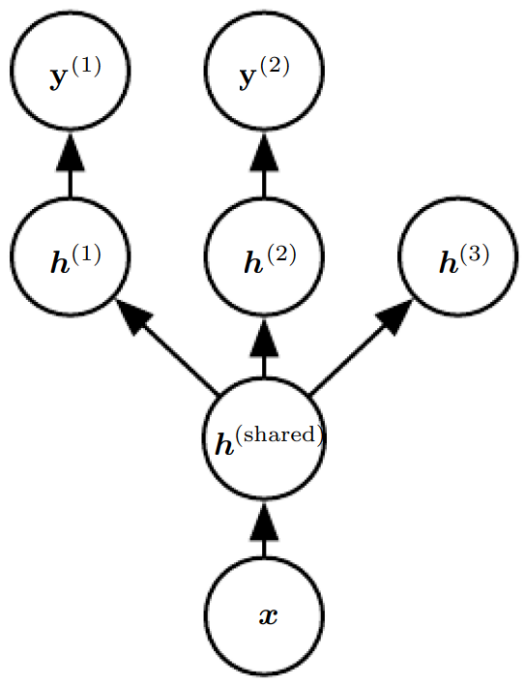


Imagen tomada de Goodfellow et al. Deep Learning, 2016

Dropout (desactivación)

- Desactiva neuronas de forma aleatoria [1] para evitar co-adaptación

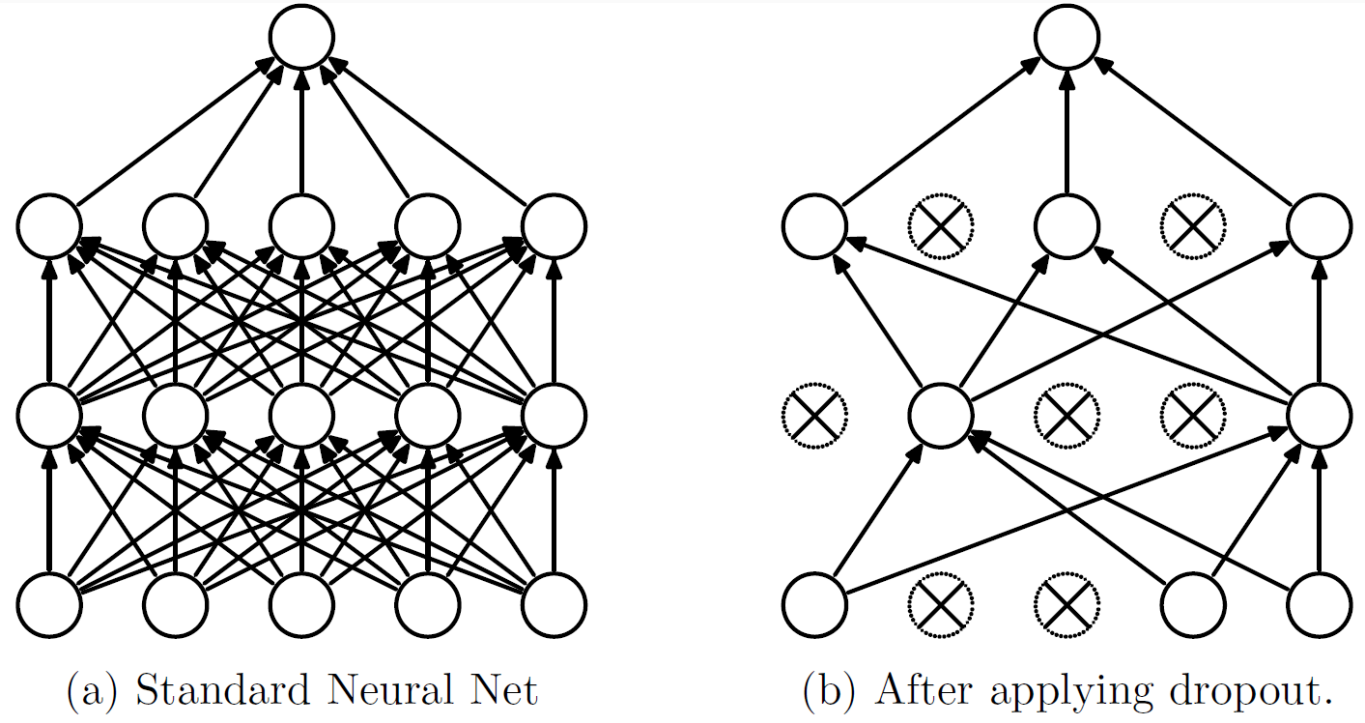


Imagen tomada de Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014

[1] Probabilidad es típicamente 0.5

Dropout en entrenamiento

- La salida de la i-ésima neurona está dada por

$$z_i^{\{\ell+1\}} = \mathbf{w}_i^{\{\ell+1\}} \widetilde{\mathbf{y}}^{\{\ell\}} + b_i^{\{\ell+1\}}$$
$$y_i^{\{\ell+1\}} = \phi(z_i^{\{\ell+1\}})$$

  donde $\widetilde{\mathbf{y}}$ es una máscara binaria sobre las salidas de las neuronas con 1s para las activas y 0s para las inactivas

$$r_j \sim Bernoulli(P)$$
$$\widetilde{\mathbf{y}}^{\{\ell\}} = \mathbf{r}^{\{\ell\}} * \mathbf{y}^{\{\ell\}}$$

- P es un hiperparámetro que indica la probabilidad de que una neurona se mantenga activa

Dropout como ensamble

- Puede verse como entrenar simultáneamente múltiples redes eliminando neuronas de una red base

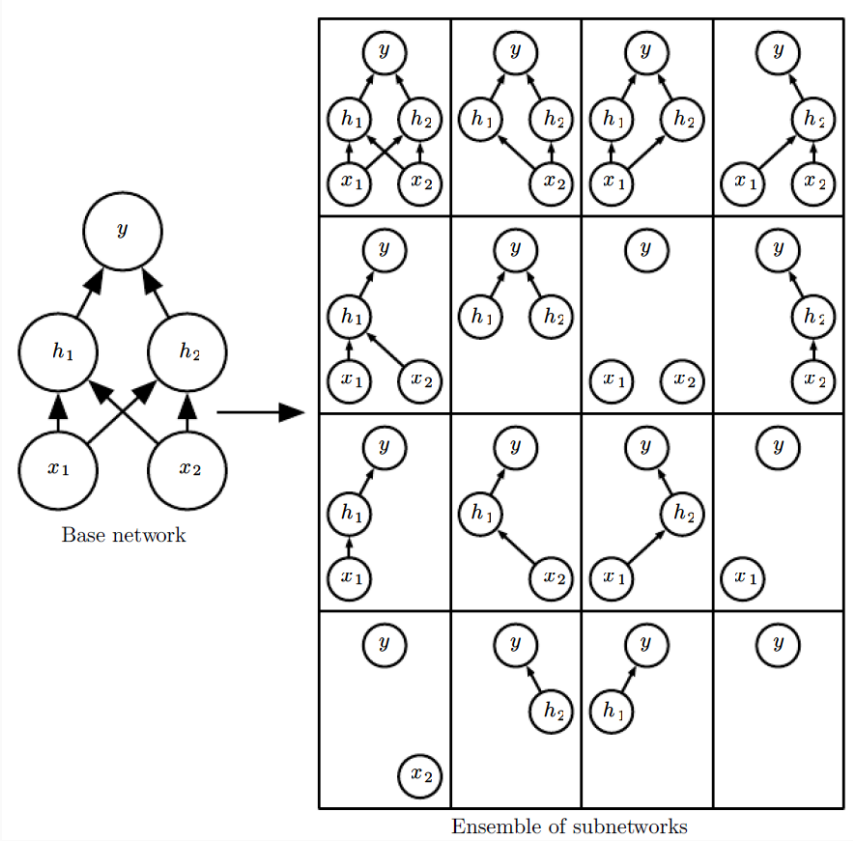


Imagen tomada de Goodfellow et al. Deep Learning, 2016

Dropout en inferencia

- En vez de promediar las salidas de todas las redes entrenadas, se obtiene la salida de una sola red con los pesos y sesgos ($\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{b}\}$) escalados

$$\boldsymbol{\theta}_{\text{inferencia}} = P \cdot \boldsymbol{\theta}$$

- De esta forma se combinan 2^n redes en una sola

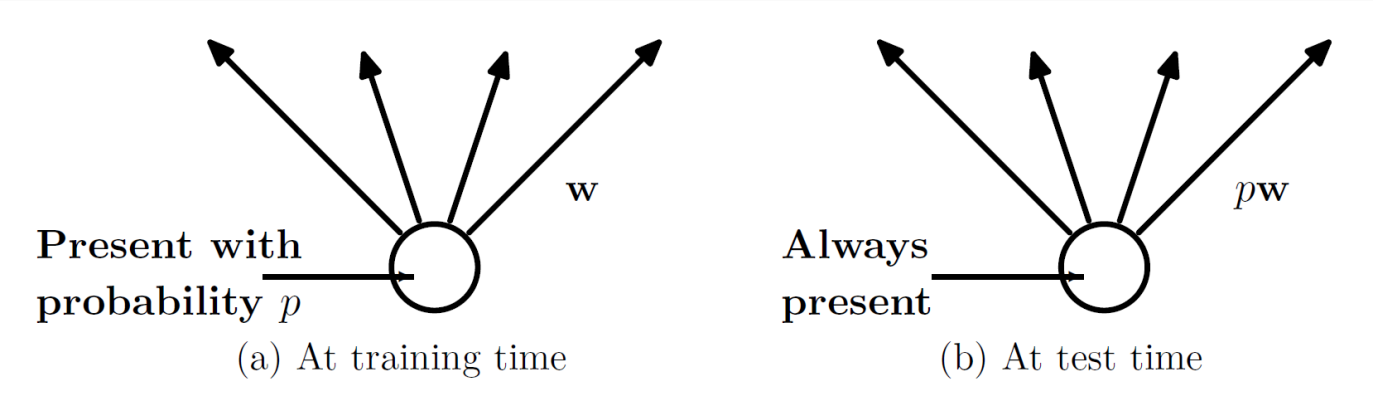


Imagen tomada de Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014

Activaciones con Dropout

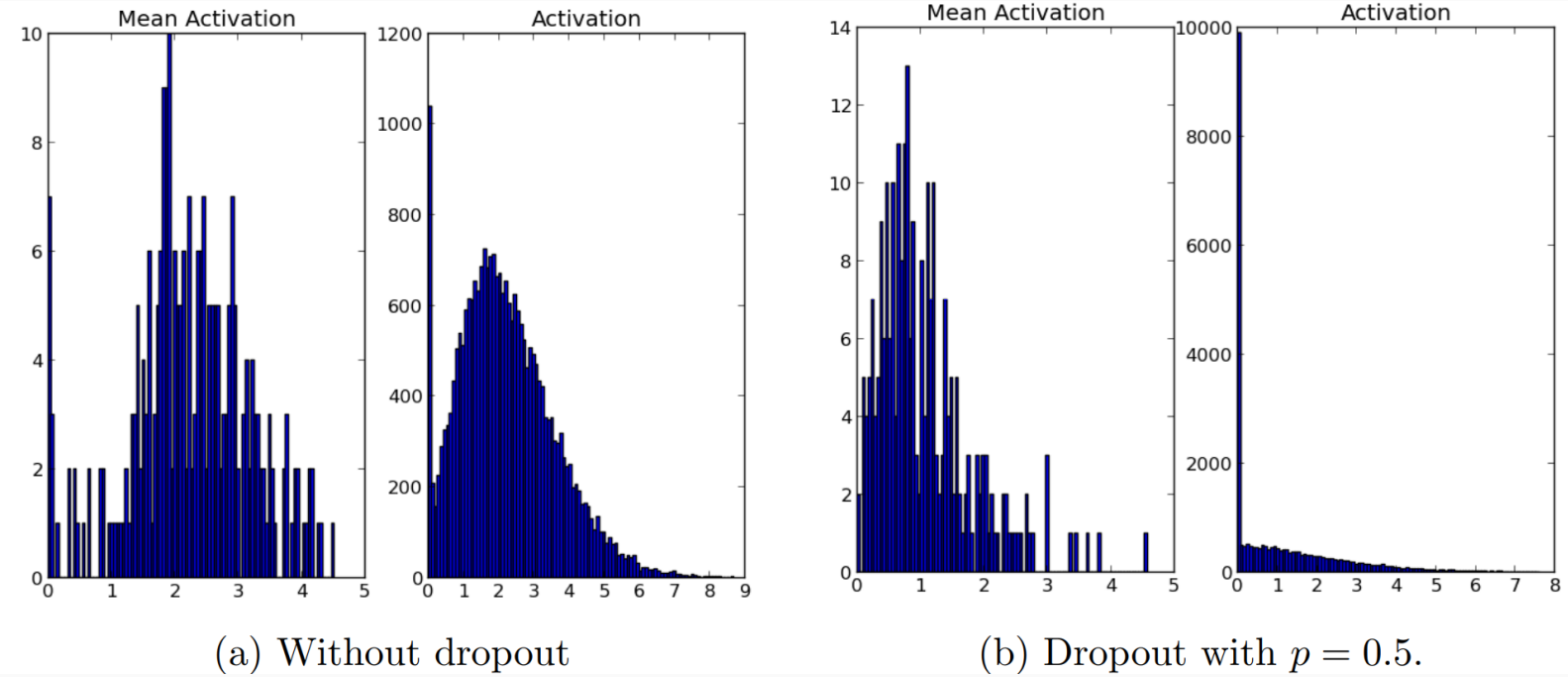
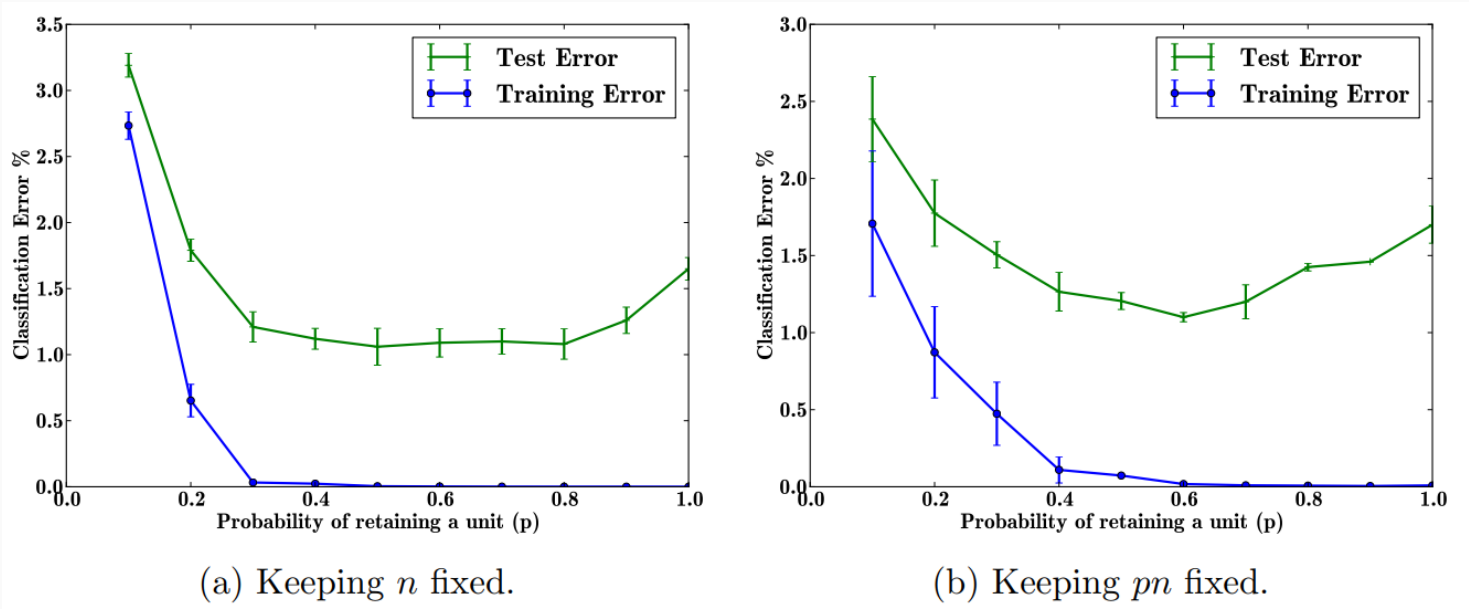


Imagen tomada de Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014

Efecto de hiperparámetro p en Dropout



(a) Keeping n fixed. (b) Keeping pn fixed.

Imagen tomada de Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014

Mejorando la convergencia

El problema del desplazamiento covariable interno

- Cambio en distribución de activaciones por cambio en parámetros durante entrenamiento

- Hace más lento el aprendizaje

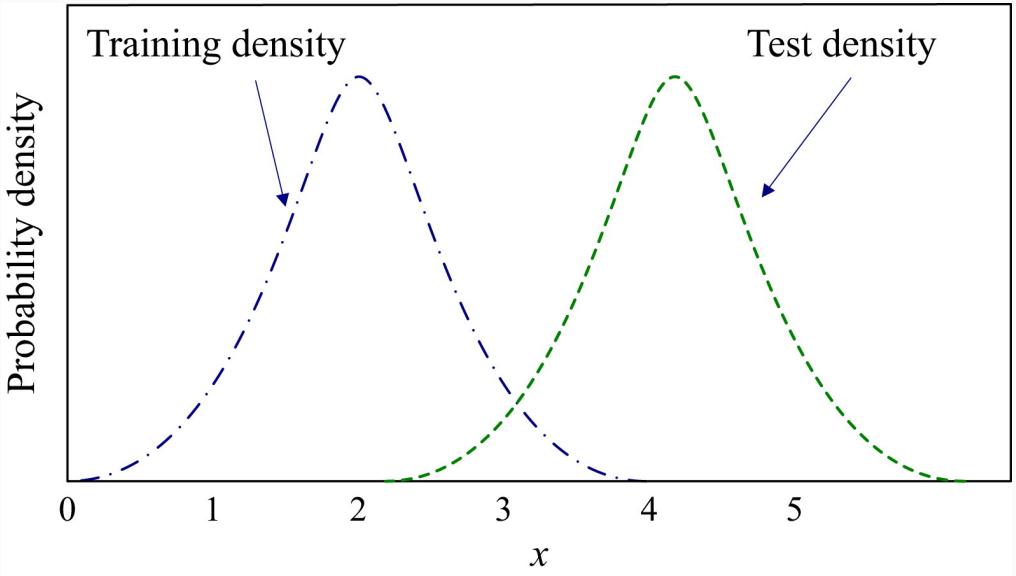


Imagen tomada de Raza et al. EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments, 2015

Normalización por lotes

- Converge más rápido si entradas tienen media 0, varianza 1 y no están correlacionadas
  1. Media del lote
  $$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x^{\{i\}}$$
  2. Varianza del lote
  $$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x^{\{i\}} - \mu_{\mathcal{B}}))^2$$
  3. Normalización
  $$\hat{x}^{\{i\}} \leftarrow \frac{x^{\{i\}} - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$$
  4. Escalado y desplazamiento
  $$y^{\{i\}} \leftarrow \gamma\hat{x}^{\{i\}} + \beta$$

Retropropagación en normalización por lotes (1)

$$\frac{\partial \mathcal{L}}{\partial \hat{x}^{\{i\}}} = \frac{\partial \mathcal{L}}{\partial y^{\{i\}}} \cdot \gamma$$

$$\frac{\partial \mathcal{L}}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \hat{x}^{\{i\}}} \cdot \left(x^{\{i\}} - \mu_{\mathcal{B}}\right) \cdot \frac{-1}{2} \left(\sigma_{\mathcal{B}}^2 + \epsilon\right)^{\frac{-3}{2}}$$

$$\frac{\partial \mathcal{L}}{\partial \mu_{\mathcal{B}}} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \hat{x}^{\{i\}}} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$\frac{\partial \mathcal{L}}{\partial x^{\{i\}}} = \frac{\partial \mathcal{L}}{\partial \hat{x}^{\{i\}}} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \mathcal{L}}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x^{\{i\}})}{m} + \frac{\partial \mathcal{L}}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

Retropropagación en normalización por lotes (2)

$$\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial y^{\{i\}}} \cdot \hat{x}^{\{i\}}$$

$$\frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial y^{\{i\}}}$$

Entrenamiento e inferencia con normalización por lotes

1. Normalizar la red con mini-lote
2. Entrenar la red con retro-propagación
3. Transformar estadísticos del lote a estadísticos de población

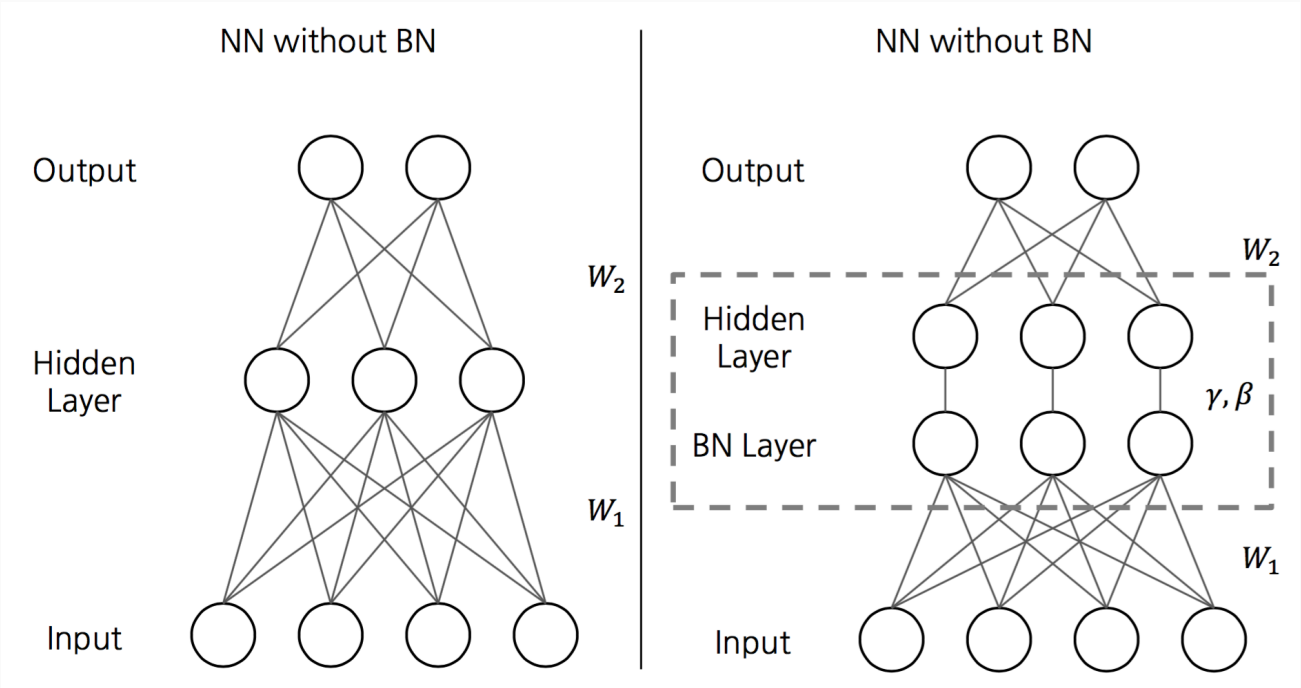


Imagen tomada de `https://wiki.tum.de/display/lfdv/Batch+Normalization`

Beneficios de normalización por lotes

- Acelera el entrenamiento
- Permite tasas de aprendizaje más grandes
- Facilita la inicialización de pesos
- Hace posible usar funciones de activación saturadas (por ej. sigmoide)
- Actúa como un tipo de regularizador
- Facilita la creación de redes profundas

Inicialización de pesos (1)

- Números aleatorios de distribución gaussiana con media 0 y varianza 0.01.
  - Funciona en redes pequeñas
  - Para redes profundas activaciones tienden a volverse 0
- Números aleatorios de distribución gaussiana con media 0 y varianza 1
  - Genera saturación de las neuronas y gradientes se vuelven 0

Inicialización de pesos (2)

- Para una capa con n_e entradas y n_s salidas
  - Uniforme de Glorot y Bengio (2010)

$$\boldsymbol{\theta} \sim \mathcal{U}\left[-\sqrt{\frac{6}{n_e + n_s}}, \sqrt{\frac{6}{n_e + n_s}}\right]$$

  - Normal de Glorot y Bengio (2010)

$$\boldsymbol{\theta} \sim \mathcal{N}\left(0, \frac{2}{n_e + n_s}\right)$$

  - Normal de He et al. (2015)

$$\boldsymbol{\theta} \sim \mathcal{N}\left(0, \frac{2}{n_e}\right)$$

Optimizadores: descenso por gradiente y variantes

Promedio móvil ponderado exponencialmente (PMPE)

- Definido por

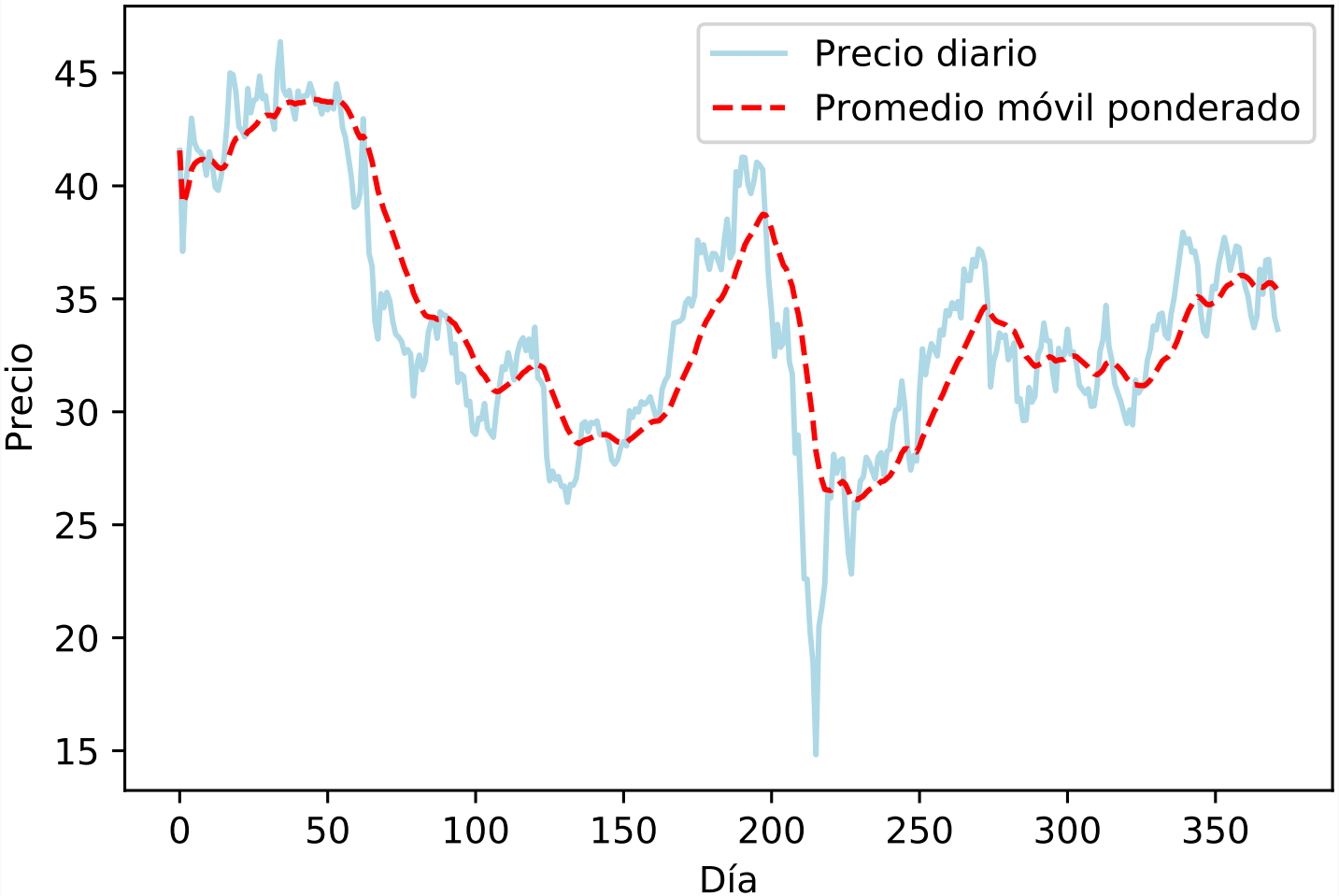
$$e^{[t+1]} = \beta \cdot e^{[t]} + (1 - \beta) \cdot v^{[t]}$$

  donde $v^{[t]}$ y $e^{[t]}$ son el valor y el PMPE en el tiempo t

- Expandiendo

$$\begin{aligned}
e^{[1]} &= v^{[1]} \\
e^{[2]} &= \beta \cdot e^{[2]} + (1 - \beta) \cdot v^{[1]} \\
e^{[3]} &= \beta \cdot e^{[3]} + (1 - \beta) \cdot v^{[2]} \\
\vdots &= \vdots \\
e^{[n]} &= \beta \cdot e^{[n]} + (1 - \beta) \cdot v^{[n-1]}
\end{aligned}$$

Promedio móvil ponderado exponencialmente (PMPE)



Recordando el descenso por gradiente estocástico

- Actualiza iterativamente los parámetros en base a los gradientes de la función de pérdida

$$\boldsymbol{\theta}^{[t+1]} = \boldsymbol{\theta}^{[t]} - \alpha \nabla \mathcal{L}(\boldsymbol{\theta}^{[t]})$$

  donde

$$\nabla \mathcal{L}(\boldsymbol{\theta}^{[t]}) = \left[\frac{\partial \mathcal{L}}{\partial \theta_0^{[t]}}, \cdots, \frac{\partial \mathcal{L}}{\partial \theta_d^{[t]}} \right]$$

- El descenso por gradiente estocástico (SGD) aproxima $\nabla \mathcal{L}(\boldsymbol{\theta}^{[t]})$ con un minilote de ejemplos de entrenamiento

SGD con momento

- Introduce término de velocidad a la actualización (acumula declive)

$$\begin{aligned}\mathbf{v}^{[t+1]} &= \mu \cdot \mathbf{v}^{[t]} - \alpha \cdot \nabla \mathcal{L}(\boldsymbol{\theta}) \\ \boldsymbol{\theta}^{[t+1]} &= \boldsymbol{\theta}^{[t]} + \mathbf{v}^{[t+1]}\end{aligned}$$

- Momento de Nesterov

$$\begin{aligned}\mathbf{v}^{[t+1]} &= \mu \cdot \mathbf{v}^{[t]} - \alpha \cdot \nabla \mathcal{L}(\boldsymbol{\theta}^{[t]} + \mu \cdot \mathbf{v}^{[t]}) \\ \boldsymbol{\theta}^{[t+1]} &= \boldsymbol{\theta}^{[t]} + \mathbf{v}^{[t+1]}\end{aligned}$$

RMSProp

- Actualiza los parámetros a partir de los promedios móviles ponderados de los gradientes al cuadrado

$$\mathbf{v}^{[t+1]} = \beta \cdot \mathbf{v}^{[t]} + (1 - \beta) \cdot \left[\nabla \mathcal{L}(\boldsymbol{\theta}^{[t]})\right]^2$$

$$\boldsymbol{\theta}^{[t+1]} = \boldsymbol{\theta}^{[t]} - \frac{\alpha}{\sqrt{\mathbf{v}^{[t+1]} + \epsilon}} \odot \nabla \mathcal{L}(\boldsymbol{\theta}^{[t]})$$

donde $\odot$ denota el producto de Hadamard

Optimizador Adam (1)

- Se estima el primer (la media) y segundo (la varianza) momentos de los gradientes

$$\mathbf{m}^{[t+1]} = \beta_1 \cdot \mathbf{m}^{[t]} + (1 - \beta_1) \cdot \nabla \mathcal{L}(\boldsymbol{\theta}^{[t]})$$

$$\mathbf{v}^{[t+1]} = \beta_2 \cdot \mathbf{v}^{[t]} + (1 - \beta_2) \cdot \left[\nabla \mathcal{L}(\boldsymbol{\theta}^{[t]})\right]^2$$

- Debido a que estas estimaciones están sesgadas hacia 0 (se inicializan con 0), se realiza una corrección

$$\hat{\mathbf{m}}^{[t+1]} = \frac{\mathbf{m}^{[t+1]}}{1 - \beta_1^{t+1}}$$

$$\hat{\mathbf{v}}^{[t+1]} = \frac{\mathbf{v}^{[t+1]}}{1 - \beta_2^{t+1}}$$

donde β_1^{t+1} y β_2^{t+1} son los factores de ponderación $\beta_1, \beta_2 \in [0, 1)$ elevados a la potencia $t + 1$

Optimizador Adam (2)

- Para actualizar los parámetros se usan las estimaciones de los momentos de los gradientes en el tiempo t

$$\boldsymbol{\theta}^{[t+1]} = \boldsymbol{\theta}^{[t]} - \frac{\alpha}{\sqrt{\hat{\mathbf{v}}^{[t+1]}} + \epsilon} \cdot \hat{\mathbf{m}}^{[t+1]}$$