

Fundamentos de Deep Learning y desarrollo de un modelo de Aprendizaje Federado para la diagnosis de COVID-19 a partir de radiografías de tórax.

Trabajo de Fin de Grado  
Francisco Miguel Castro Macías

Tutores:  
Francisco Herrera Triguero  
Siham Tabik

# Índice

## 1 Introducción

- Motivación y descripción del problema
- Objetivos

## 2 Matemáticas

- Teoría de la probabilidad
- Teoría de la información
- Optimización basada en gradiente descendente

## 3 Informática teórica

- Aprendizaje automático
- Aprendizaje federado
- Aprendizaje profundo

## 4 Informática práctica

- COVIDGR-FL
- SDNETLearning
- Experimentación

## 5 Conclusiones y vías futuras

## 1 Introducción

- Motivación y descripción del problema
- Objetivos

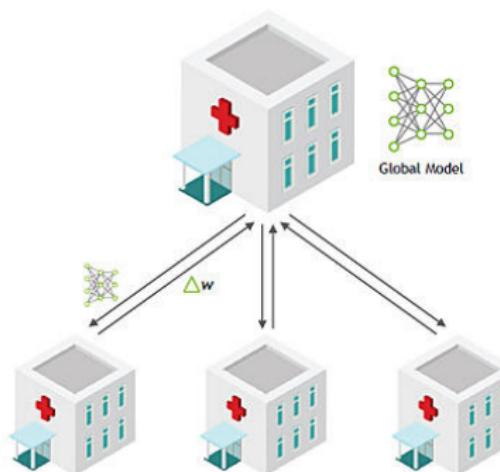
## 2 Matemáticas

## 3 Informática teórica

## 4 Informática práctica

## 5 Conclusiones y vías futuras

## Motivación y descripción del problema



- Diagnóstico de COVID-19  $\xrightarrow{\text{Compromisos de tiempo y costes}}$  Radiografías de tórax
- Clasificación de imágenes → Aprendizaje profundo
- Información de hospitales  $\xrightarrow{\text{Alta privacidad}}$  Aprendizaje federado

## Objetivos

1. Diseñar una metodología (FederatedSDNET) para detectar la COVID-19 a partir de radiografías de tórax en un escenario distribuido manteniendo la privacidad de los datos.
2. Desarrollar una librería (SDNETLearning) que permita su entrenamiento y la inferencia de nuevos diagnósticos en un escenario federado.
3. Construir un nuevo conjunto de datos (COVIDGR-FL) para particionarlo en varios nodos de acuerdo a diversos criterios.
4. Analizar el comportamiento de COVID-SDNET en el nuevo conjunto de datos.
5. Analizar la escalabilidad de la metodología.
6. Comparar el comportamiento de dos operadores de agregación de parámetros.

## 1 Introducción

## 2 Matemáticas

- Teoría de la probabilidad
- Teoría de la información
- Optimización basada en gradiente descendente

## 3 Informática teórica

## 4 Informática práctica

## 5 Conclusiones y vías futuras

### ¿En qué se sostenta el aprendizaje?

1. Teoría de la probabilidad.
2. Teoría de la información.
3. Optimización basada en gradiente descendente.

# Divergencias

## Definición (*Divergencia*)

Sea  $X$  un conjunto. Una divergencia es una aplicación  $D: X \times X \rightarrow \mathbb{R}$  que cumple

1. Para cualesquiera  $x, y \in X$  se cumple  $D(x, y) \geq 0$ .
2.  $D(x, y) = 0$  si y sólo si  $x = y$ .

## Definición

- Divergencia de Kullback-Leibler:

$$D_{KL}(P, Q) = \mathbb{E}_{X \sim P} \left[ \log \left( \frac{p(X)}{q(X)} \right) \right]$$

- Divergencia de Jensen-Shannon:

$$D_{JS}(P, Q) = \frac{1}{2} (D_{KL}(P, M) + D_{KL}(Q, M)), \quad M = \frac{P+Q}{2}$$

# Divergencias

## Definición (*Divergencia*)

Sea  $X$  un conjunto. Una divergencia es una aplicación  $D: X \times X \rightarrow \mathbb{R}$  que cumple

1. Para cualesquiera  $x, y \in X$  se cumple  $D(x, y) \geq 0$ .
2.  $D(x, y) = 0$  si y sólo si  $x = y$ .

## Definición

- Divergencia de Kullback-Leibler:

$$D_{KL}(P, Q) = \mathbb{E}_{X \sim P} \left[ \log \left( \frac{p(X)}{q(X)} \right) \right]$$

- Divergencia de Jensen-Shannon:

$$D_{JS}(P, Q) = \frac{1}{2} (D_{KL}(P, M) + D_{KL}(Q, M)), \quad M = \frac{P + Q}{2}$$

## Divergencias

Sean  $P$  y  $Q$  dos medidas de probabilidad en  $\Omega \subset \mathbb{R}^n$  de las que se conocen funciones de densidad,  $p$  y  $q$  respectivamente.

## **Teorema (*Desigualdad de la información*)**

- $D_{KL}(P, Q) \geq 0$
  - $D_{KL}(P, Q) = 0$  si y sólamente si  $p = q$  en casi todo punto.

## Corolario

- $D_{JS}(P, Q) \geq 0$
  - $D_{JS}(P, Q) = 0$  si y sólamente si  $p = q$  en casi todo punto.

## Entropía y entropía cruzada

## Definición (*Entropía*)

$$H(P) = -\mathbb{E}_{X \sim P} [\log(p(X))]$$

## Entropía y entropía cruzada

## Definición (*Entropía*)

$$H(P) = -\mathbb{E}_{X \sim P} [\log(p(X))]$$

## Definición (*Entropía cruzada*)

$$H(P, Q) = H(P) + D_{KL}(P, Q) = -\mathbb{E}_{X \sim P} [\log(q(X))]$$

# Entropía y entropía cruzada

## Definición (*Entropía*)

$$H(P) = -\mathbb{E}_{X \sim P} [\log(p(X))]$$

## Definición (*Entropía cruzada*)

$$H(P, Q) = H(P) + D_{KL}(P, Q) = -\mathbb{E}_{X \sim P} [\log(q(X))]$$

## Observación

Sea  $\mathcal{H} = \{f(x, \alpha) : \alpha \in \Lambda\}$  un conjunto de funciones de densidad parametrizadas. Entonces

$$J(\alpha) = H(P, F_\alpha) = -\mathbb{E}_{X \sim P} [\log(f(X, \alpha))], \quad \forall \alpha \in \Lambda$$

Si  $\Lambda \subset \mathbb{R}^d$  encontrar la distribución de  $\mathcal{H}$  más parecida a  $P$  se corresponde con un problema de optimización real.

# Funciones convexas y desigualdad de Jensen

## Teorema (*Desigualdad de Jensen*)

Sea  $(\Omega, \mathcal{A}, P)$  un espacio de probabilidad,  $\varphi: ]a, b[ \rightarrow \mathbb{R}$  una función convexa y  $f: \Omega \rightarrow \mathbb{R}$  es una función  $P$ -integrable con  $f(\Omega) \subset ]a, b[$ . Entonces:

$$\varphi \left( \int_{\Omega} f \, dP \right) \leq \int_{\Omega} (\varphi \circ f) \, dP$$

Además, la igualdad se da, si y sólo si, o bien  $f$  es constante, o bien  $\varphi$  coincide con una función lineal en casi todo punto de  $f(\Omega)$ .

## Lema

Sea  $f: ]a, b[ \rightarrow \mathbb{R}$  convexa. Sea  $c \in ]a, b[$ . Existe  $r: \mathbb{R} \rightarrow \mathbb{R}$  lineal cumpliendo  $r(c) = f(c)$  y  $r(x) \leq f(x)$  para cada  $x \in ]a, b[$ .

# Funciones convexas y desigualdad de Jensen

## Teorema (*Desigualdad de Jensen*)

Sea  $(\Omega, \mathcal{A}, P)$  un espacio de probabilidad,  $\varphi: ]a, b[ \rightarrow \mathbb{R}$  una función convexa y  $f: \Omega \rightarrow \mathbb{R}$  es una función  $P$ -integrable con  $f(\Omega) \subset ]a, b[$ . Entonces:

$$\varphi \left( \int_{\Omega} f \, dP \right) \leq \int_{\Omega} (\varphi \circ f) \, dP$$

Además, la igualdad se da, si y sólo si, o bien  $f$  es constante, o bien  $\varphi$  coincide con una función lineal en casi todo punto de  $f(\Omega)$ .

## Lema

Sea  $f: ]a, b[ \rightarrow \mathbb{R}$  convexa. Sea  $c \in ]a, b[$ . Existe  $r: \mathbb{R} \rightarrow \mathbb{R}$  lineal cumpliendo  $r(c) = f(c)$  y  $r(x) \leq f(x)$  para cada  $x \in ]a, b[$ .

# Gradiente descendente

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

## Proposición

Sea  $f: \mathbb{R}^d \rightarrow \mathbb{R}$  tal que  $f \in C^1(\mathbb{R}^d)$ , es convexa y su gradiente es lipschitziano con constante  $L \geq 0$ :

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^d$$

Supongamos también que  $x^*$  es un mínimo local de  $f$  y fijemos  $\eta \leq \frac{1}{L}$  y  $x_0 \in \mathbb{R}^d$ . Entonces:

$$f(x_k) - f(x^*) \leq \frac{\|x_0 - x^*\|^2}{2\eta k}$$

## Gradiente descendente

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

### Proposición

Sea  $f: \mathbb{R}^d \rightarrow \mathbb{R}$  tal que  $f \in C^1(\mathbb{R}^d)$ , es convexa y su gradiente es lipschitziano con constante  $L \geq 0$ :

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^d$$

Supongamos también que  $x^*$  es un mínimo local de  $f$  y fijemos  $\eta \leq \frac{1}{L}$  y  $x_0 \in \mathbb{R}^d$ . Entonces:

$$f(x_k) - f(x^*) \leq \frac{\|x_0 - x^*\|^2}{2\eta k}$$

## 1 Introducción

2 Matemáticas

### 3 Informática teórica

- Aprendizaje automático
  - Aprendizaje federado
  - Aprendizaje profundo

## 4 Informática práctica

## 5 Conclusiones y vías futuras

## Aprendizaje estadístico

- $\mathcal{X} \times \mathcal{Y}$ , existe  $P$  distribución de probabilidad sobre  $\mathcal{X} \times \mathcal{Y}$ .
  - $\Lambda \subset \mathbb{R}^d$  conjunto de parámetros,  $\mathcal{H} = \{f(x, \alpha) : \alpha \in \Lambda\}$ .
  - $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$  error puntual.
  - $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$  realización de una m.a.s. de  $(X, Y) \sim P$ .

## Aprendizaje estadístico

- $\mathcal{X} \times \mathcal{Y}$ , existe  $P$  distribución de probabilidad sobre  $\mathcal{X} \times \mathcal{Y}$ .
  - $\Lambda \subset \mathbb{R}^d$  conjunto de parámetros,  $\mathcal{H} = \{f(x, \alpha) : \alpha \in \Lambda\}$ .
  - $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$  error puntual.
  - $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$  realización de una m.a.s. de  $(X, Y) \sim P$ .

## Medidas de error

- Error fuera de la muestra:

$$L(\alpha) = \mathbb{E} [\ell(Y, f(X, \alpha))] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, f(x, \alpha)) dP(x, y), \quad \alpha \in \Lambda$$

- #### ■ Error empírico o de entrenamiento:

## Aprendizaje estadístico

- $\mathcal{X} \times \mathcal{Y}$ , existe  $P$  distribución de probabilidad sobre  $\mathcal{X} \times \mathcal{Y}$ .
  - $\Lambda \subset \mathbb{R}^d$  conjunto de parámetros,  $\mathcal{H} = \{f(x, \alpha) : \alpha \in \Lambda\}$ .
  - $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$  error puntual.
  - $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$  realización de una m.a.s. de  $(X, Y) \sim P$ .

## Medidas de error

- Error fuera de la muestra:

$$L(\alpha) = \mathbb{E} [\ell(Y, f(X, \alpha))] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, f(x, \alpha)) dP(x, y), \quad \alpha \in \Lambda$$

- #### ■ Error empírico o de entrenamiento:

$$L_{emp}(\alpha) = \frac{1}{N} \sum_{n=1}^N \ell(y_n, f(x_n, \alpha)), \quad \alpha \in \Lambda$$

# Desigualdad de Vapnik-Chervonenkis

Para cada  $\epsilon > 0$  se cumple:

$$\mathbb{P} \left[ \sup_{\alpha \in \Lambda} |L_{emp}(\alpha) - L(\alpha)| > \epsilon \right] \leq \Gamma(N, d_{VC}(\mathcal{H}), \epsilon)$$

## Propiedades de $\Gamma$

- $\lim_{N \rightarrow \infty} \Gamma(N, d_{VC}(\mathcal{H}), \epsilon) = 0$ .
- Si  $d_{VC}(\mathcal{H})$  crece, entonces  $\Gamma(N, d_{VC}(\mathcal{H}), \epsilon)$  crece.

## Conclusiones

- El aprendizaje es factible.
- Cuando más complejo es el modelo más datos necesitamos para mantener el error de generalización bajo.

# Desigualdad de Vapnik-Chervonenkis

Para cada  $\epsilon > 0$  se cumple:

$$\mathbb{P} \left[ \sup_{\alpha \in \Lambda} |L_{emp}(\alpha) - L(\alpha)| > \epsilon \right] \leq \Gamma(N, d_{VC}(\mathcal{H}), \epsilon)$$

## Propiedades de $\Gamma$

- $\lim_{N \rightarrow \infty} \Gamma(N, d_{VC}(\mathcal{H}), \epsilon) = 0$ .
- Si  $d_{VC}(\mathcal{H})$  crece, entonces  $\Gamma(N, d_{VC}(\mathcal{H}), \epsilon)$  crece.

## Conclusiones

- El aprendizaje es factible.
- Cuando más complejo es el modelo más datos necesitamos para mantener el error de generalización bajo.

## Desigualdad de Vapnik-Chervonenkis

Para cada  $\epsilon > 0$  se cumple:

$$\mathbb{P} \left[ \sup_{\alpha \in \Lambda} |L_{emp}(\alpha) - L(\alpha)| > \epsilon \right] \leq \Gamma(N, d_{VC}(\mathcal{H}), \epsilon)$$

### Propiedades de $\Gamma$

- $\lim_{N \rightarrow \infty} \Gamma(N, d_{VC}(\mathcal{H}), \epsilon) = 0$ .
- Si  $d_{VC}(\mathcal{H})$  crece, entonces  $\Gamma(N, d_{VC}(\mathcal{H}), \epsilon)$  crece.

### Conclusiones

- El aprendizaje es factible.
- Cuando más complejo es el modelo más datos necesitamos para mantener el error de generalización bajo.

## Escenario distribuido/federado

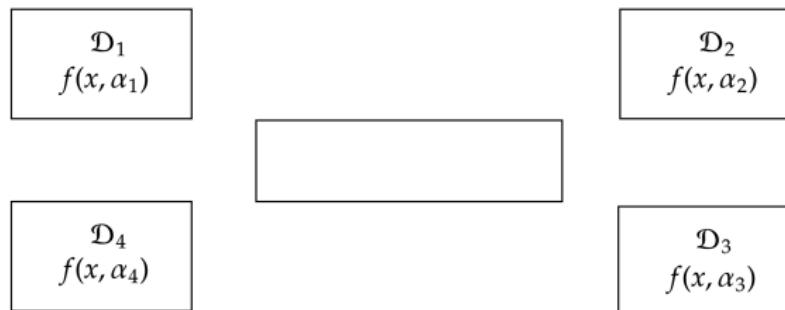
¡Datos repartidos en varios clientes!

- $\mathcal{D}_k$  realización de una m.a.s. generada de acuerdo a  $P_k$
- Queremos resolver:  $\min_{\alpha \in \Lambda} \frac{1}{K} \sum_{k=1}^K L_k(\alpha)$
- Optimización federada: cada nodo entrena y comparte los parámetros que ha obtenido. Estos se agregan mediante un operador de agregación  $\Delta: \Lambda^K \rightarrow \Lambda$ .

## Escenario distribuido/federado

¡Datos repartidos en varios clientes!

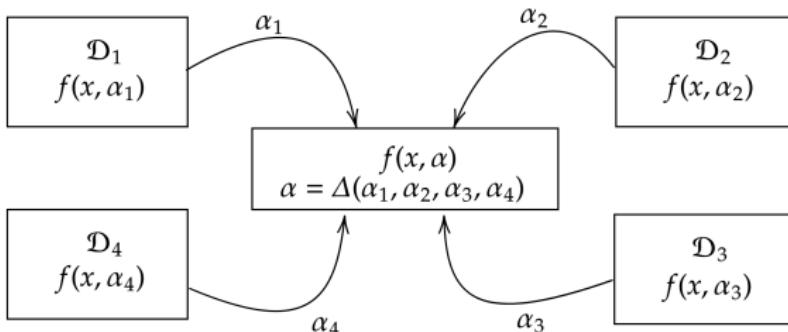
- $\mathcal{D}_k$  realización de una m.a.s. generada de acuerdo a  $P_k$
- Queremos resolver:  $\min_{\alpha \in \Lambda} \frac{1}{K} \sum_{k=1}^K L_k(\alpha)$
- Optimización federada: cada nodo entrena y comparte los parámetros que ha obtenido. Estos se agregan mediante un operador de agregación  $\Delta: \Lambda^K \rightarrow \Lambda$ .



# Escenario distribuido/federado

**¡Datos repartidos en varios clientes!**

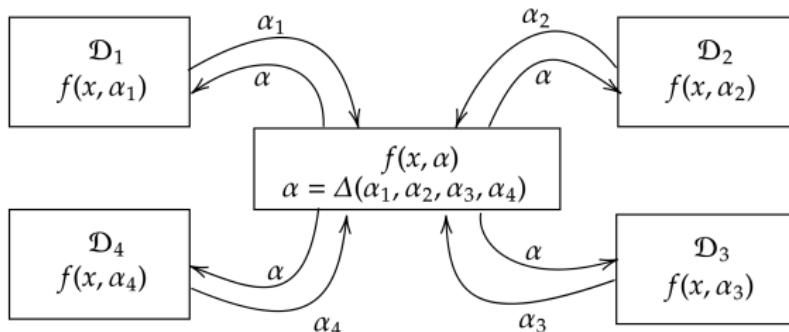
- $\mathcal{D}_k$  realización de una m.a.s. generada de acuerdo a  $P_k$
- Queremos resolver:  $\min_{\alpha \in \Lambda} \frac{1}{K} \sum_{k=1}^K L_k(\alpha)$
- Optimización federada: cada nodo entrena y comparte los parámetros que ha obtenido. Estos se agregan mediante un operador de agregación  $\Delta: \Lambda^K \rightarrow \Lambda$ .



# Escenario distribuido/federado

**¡Datos repartidos en varios clientes!**

- $\mathcal{D}_k$  realización de una m.a.s. generada de acuerdo a  $P_k$
- Queremos resolver:  $\min_{\alpha \in \Lambda} \frac{1}{K} \sum_{k=1}^K L_k(\alpha)$
- Optimización federada: cada nodo entrena y comparte los parámetros que ha obtenido. Estos se agregan mediante un operador de agregación  $\Delta: \Lambda^K \rightarrow \Lambda$ .

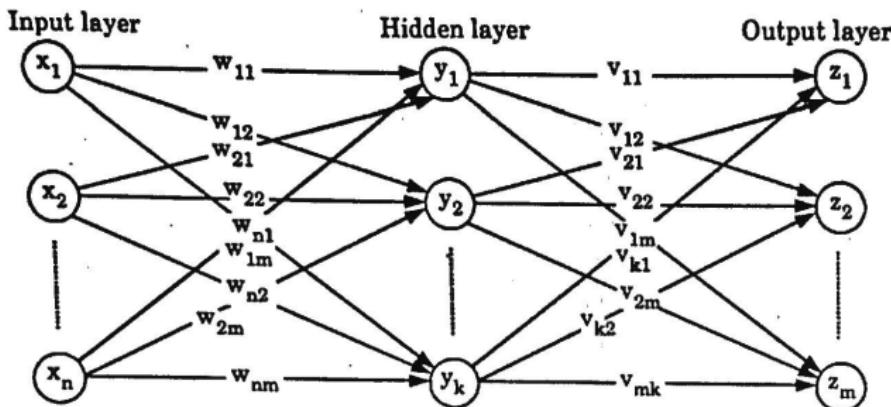


# Redes neuronales pre-alimentadas (FFNN)

## Definición

- Para cada  $t: d_t \in \mathbb{N}: \theta_t: \mathbb{R}^{d_t} \rightarrow \mathbb{R}^{d_{t+1}}$ ,  $W_t \in \mathbb{R}^{d_t \times d_{t-1}}$ .
- $h(x, W) = x_T$  dada por

$$\begin{cases} x_0 = x \in \mathbb{R}^{d_0}, \\ x_t = \theta_t(W_t x_{t-1}), \quad t \in \{1, \dots, T\} \end{cases}$$



# FFNN: entrenamiento

## Propagación hacia delante

$$x_0 \rightarrow x_1 \rightarrow \cdots \rightarrow x_{t-1} \rightarrow x_t \rightarrow \cdots \rightarrow x_{T-1} \rightarrow x_T$$

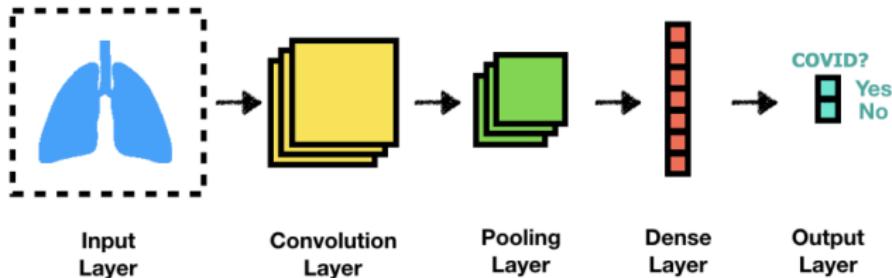
## Propagación hacia atrás

Si  $\ell: \mathbb{R}^{d_T} \rightarrow \mathbb{R}$  es diferenciable y cada  $\theta_t \in C^1$ , podemos calcular los gradientes usando las sensibilidades  $\delta_t$ :

$$\delta_T \rightarrow \delta_{T-1} \rightarrow \cdots \rightarrow \delta_{t+1} \rightarrow \delta_t \rightarrow \cdots \rightarrow \delta_2 \rightarrow \delta_1$$

A partir de cada  $\delta_t$  calculamos  $\nabla_{W_t} \ell(h(x, W))$ .

# Redes neuronales convolucionales (CNNs)



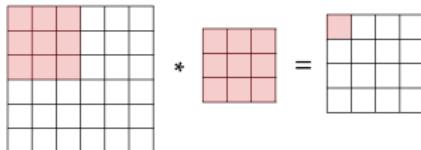
# CNNs: convolución discreta

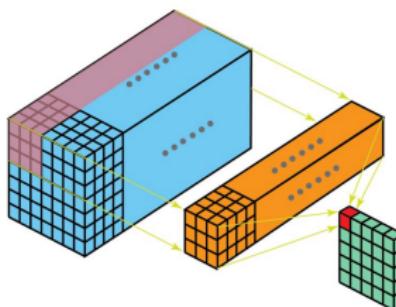
## Convolución discreta

$$(F, G) \mapsto F * G$$

$$(F * G)_{i,j} = \sum_{n,m=-b}^b F_{i-n,j-m} G_{n,m}$$

$$(F * G)_{i,j} = \sum_{k=1}^c (F(k) * G(k))_{i,j}$$


$$\begin{matrix} & * & = & \end{matrix}$$

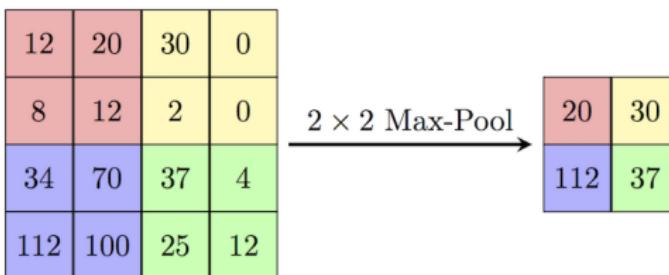


# CNNs: pooling

## Pooling

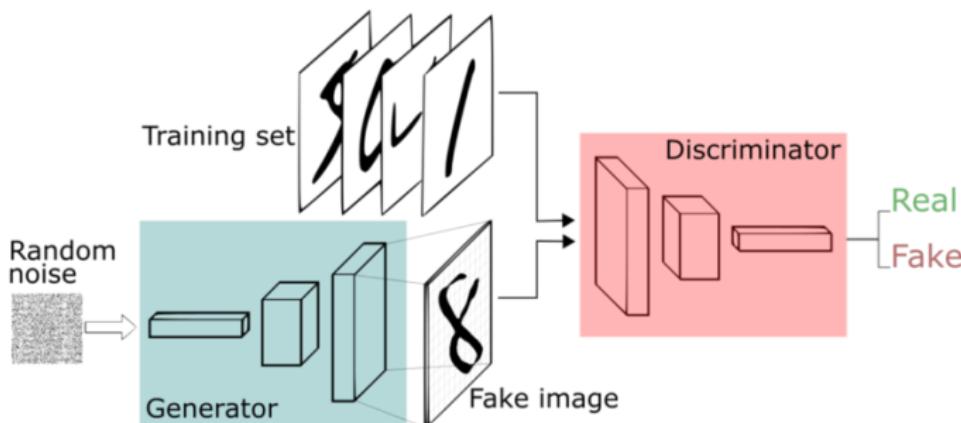
$$M \in \mathbb{R}^{sa \times sb} \mapsto Pool(M) \in \mathbb{R}^{a \times b}$$

Pool actúa sobre cada submatriz  $s \times s$  de  $M$  mediante  $f_{Pool}: \mathbb{R}^{s \times s} \rightarrow \mathbb{R}$ .



# Redes generativas antagónicas (GANs)

- Generador  $G: \mathcal{Z} \rightarrow \mathcal{X}$  genera nuevos ejemplos intentando engañar al clasificador.
- Discriminador  $D: \mathcal{X} \rightarrow [0, 1]$ ,  $D(x)$  es la probabilidad de que un ejemplo  $x$  sea real.



## GANs: formulación y resultados

$P_r$  distribución en  $\mathcal{X}$ ,  $P_z$  distribución en  $\mathcal{Z}$ ,  $P_G = P_z \circ G^{-1}$  distribución inducida por el generador.

$$V(G, D) = \mathbb{E}_{X \sim P_r} [\log(D(X))] + \mathbb{E}_{Z \sim P_z} [\log(1 - D(G(Z)))]$$

$$\min_G \max_D V(G, D)$$

### Proposición

1. *El discriminador óptimo  $D^*$  viene dado por:*

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_G(x)}$$

2. *Fijado el discriminador óptimo  $D^*$ , el generador óptimo  $G^*$  se alcanza sólo cuando  $p_{G^*} = p_r$ .*

## GANs: formulación y resultados

$P_r$  distribución en  $\mathcal{X}$ ,  $P_z$  distribución en  $\mathcal{Z}$ ,  $P_G = P_z \circ G^{-1}$  distribución inducida por el generador.

$$V(G, D) = \mathbb{E}_{X \sim P_r} [\log(D(X))] + \mathbb{E}_{Z \sim P_z} [\log(1 - D(G(Z)))]$$

$$\min_G \max_D V(G, D)$$

### Proposición

1. *El discriminador óptimo  $D^*$  viene dado por:*

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_G(x)}$$

2. *Fijado el discriminador óptimo  $D^*$ , el generador óptimo  $G^*$  se alcanza sólo cuando  $p_{G^*} = p_r$ .*

# FUCITNET

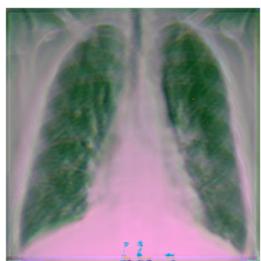
- Diseñado para problemas de clasificación:  $\mathcal{Y} = \{1, \dots, C\}$ .
- Conjunto de generadores  $\{G_k: \mathcal{X} \rightarrow \mathcal{X}: k \in \mathcal{Y}\}$ . Entrenados para minimizar la versión empírica de:

$$\underbrace{l_{MSE}(k) + 0,006l_{perceptual}(k)}_{\text{Término de similaridad}} + \lambda l_{CE}(k).$$

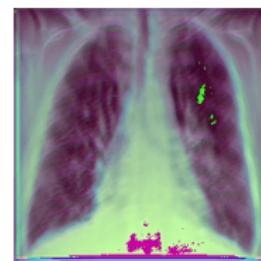
- Clasificador  $D: \mathcal{X} \rightarrow [0, 1]^C$ ,  $D(x)_k$  pretende aproximar  $P(y = k|x)$ . Entrenado para minimizar la entropía cruzada  $l_{CE}$  de las imágenes transformadas.



(a) Imagen original  
 $x(N)$

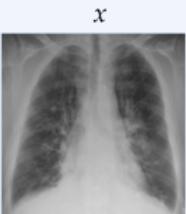
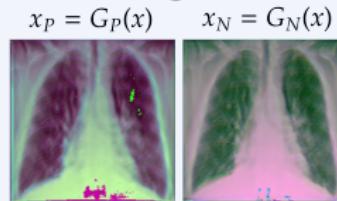
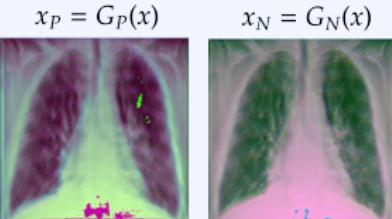


(b) Transformación negativa  $G_N(x)$



(c) Transformación positiva  $G_P(x)$

# La metodología SDNET

**FASE 1**Segmentación  
+  
Recorte**FASE 2**FUCITNET  
 $G = (G_P, G_N)$ **FASE 3**

Clasificador RESNET-50

$$\Theta = [\theta_{PTP}, \theta_{PTN}, \theta_{NTP}, \theta_{NTN}]$$

$$y_P = \Theta(x_P), y_N = \Theta(x_N) \in [0, 1]^4$$

Inferencia

$$\Phi$$

$$\Phi(y_P, y_N) = y \in \{P, N\}$$

**FederatedSDNET** → resultado de aplicar el esquema de optimización federada a la metodología SDNET.

## 1 Introducción

## 2 Matemáticas

## 3 Informática teórica

## 4 Informática práctica

- COVIDGR-FL
- SDNETLearning
- Experimentación

## 5 Conclusiones y vías futuras

COVIDGR-FL

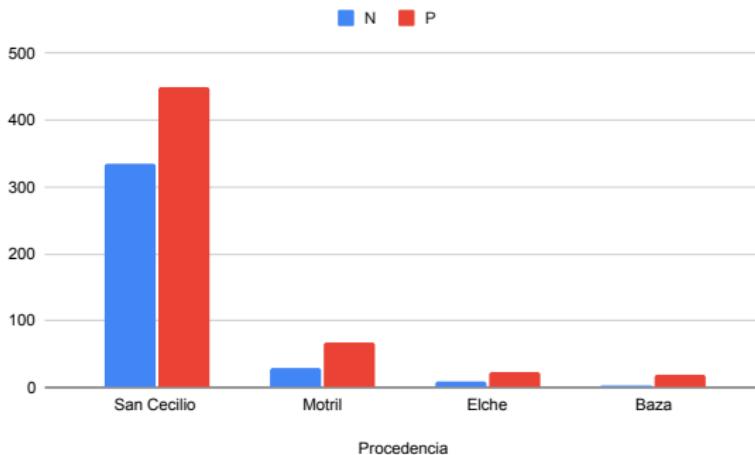
## Conjunto de datos diseñado

- Imágenes etiquetadas por radiólogos de 4 hospitales distintos.
  - 490 imágenes clase negativa y 450 imágenes clase positiva.
  - Características: Procedencia, edad/sexo, puntos RALE, niveles RALE.

# COVIDGR-FL

## Conjunto de datos diseñado

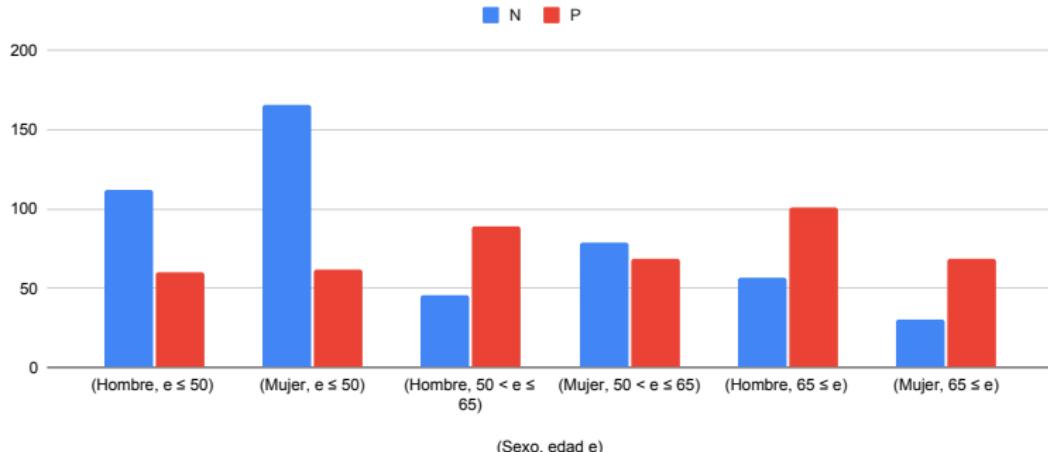
- Imágenes etiquetadas por radiólogos de 4 hospitales distintos.
- 490 imágenes clase negativa y 450 imágenes clase positiva.
- Características: **Procedencia**, edad/sexo, puntos RALE, niveles RALE.



# COVIDGR-FL

## Conjunto de datos diseñado

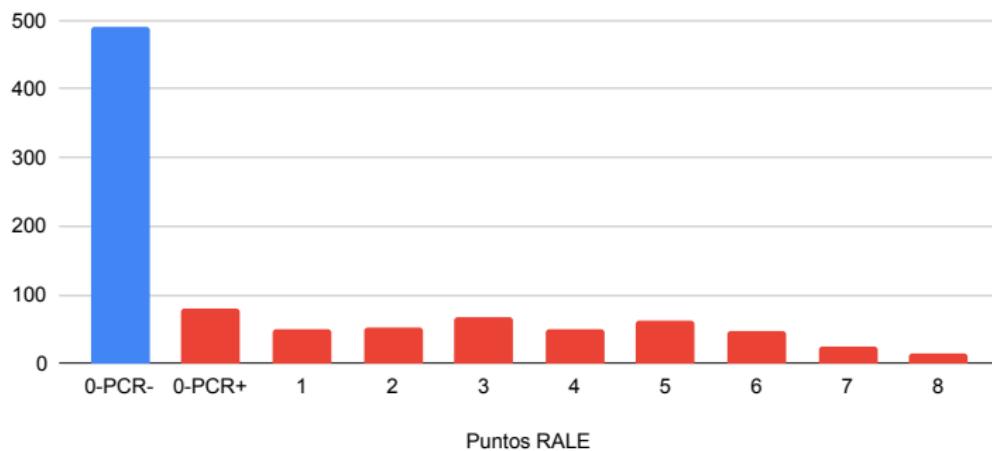
- Imágenes etiquetadas por radiólogos de 4 hospitales distintos.
- 490 imágenes clase negativa y 450 imágenes clase positiva.
- Características: Procedencia, **edad/sexo**, puntos RALE, niveles RALE.



# COVIDGR-FL

## Conjunto de datos diseñado

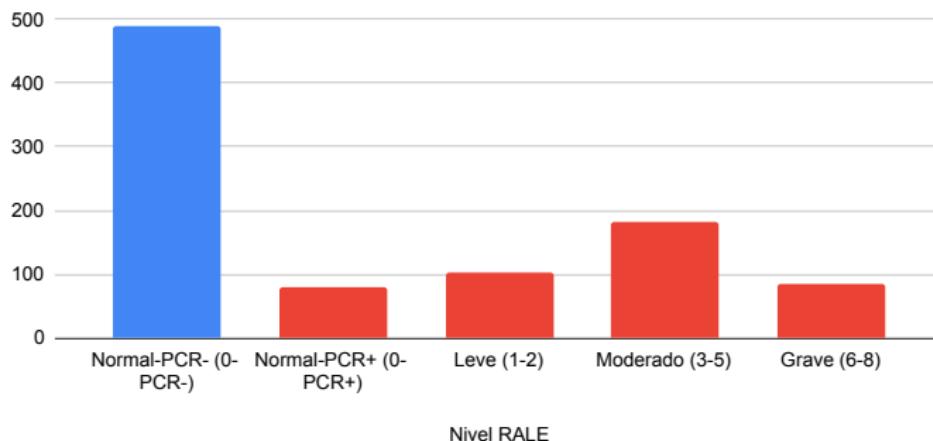
- Imágenes etiquetadas por radiólogos de 4 hospitales distintos.
- 490 imágenes clase negativa y 450 imágenes clase positiva.
- Características: Procedencia, edad/sexo, **puntos RALE**, niveles RALE.



COVIDGR-FL

## Conjunto de datos diseñado

- Imágenes etiquetadas por radiólogos de 4 hospitales distintos.
  - 490 imágenes clase negativa y 450 imágenes clase positiva.
  - Características: Procedencia, edad/sexo, puntos RALE, **niveles RALE**.



# SDNELLearning

## Librería desarrollada

- Implementa el modelo FUCITNET. Ofrece la posibilidad de entrenar y evaluar el par (*Generadores, Clasificador*) para un problema de clasificación arbitrario.
- Implementa la metodología SDNET en el escenario centralizado. Proporciona las funcionalidades necesarias para llevar a cabo cada una de las cuatro etapas.
- Implementa la metodología SDNET en el escenario federado. Permite entrenar el modelo e inferir diagnósticos en configuraciones distribuidas.

¿Sobre qué está construida?



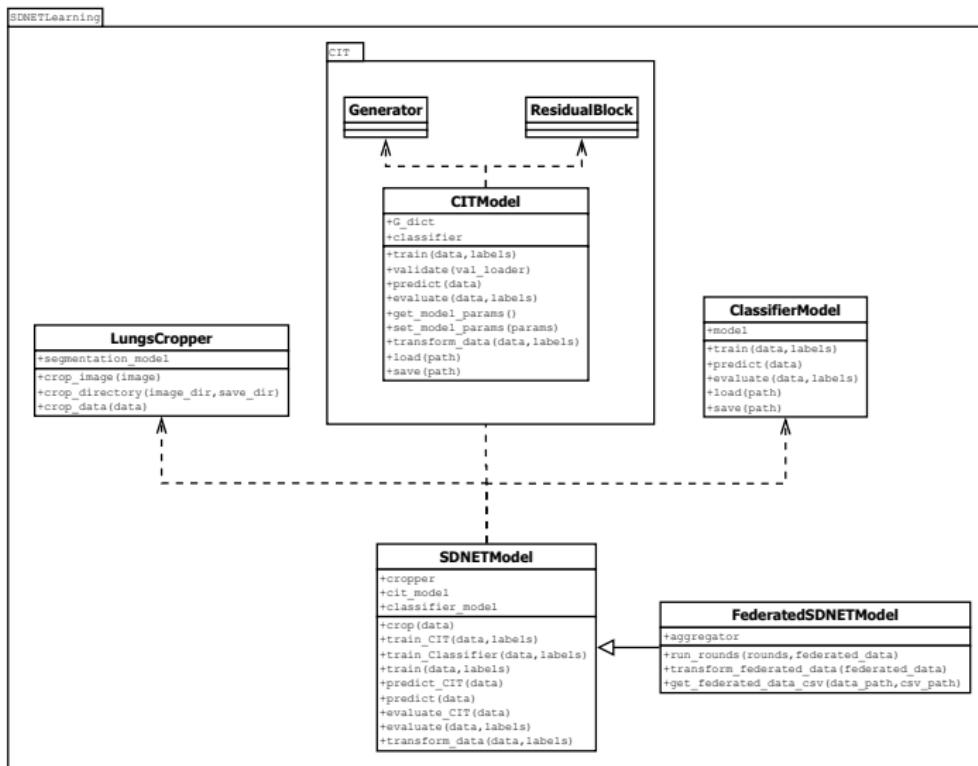
TensorFlow



PyTorch

sherpa.ai

# SDNETLearning: diagrama de clases



# Experimentación

## Operadores de agregación

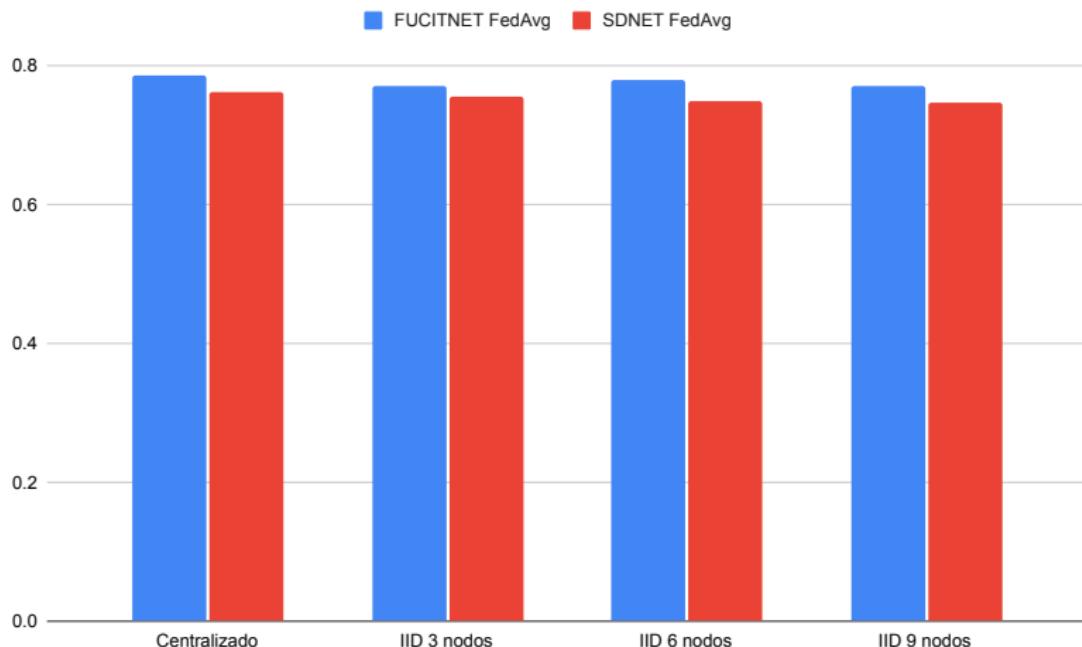
$$\Delta(\alpha_1, \dots, \alpha_K) = \sum_{k=1}^K w_k \alpha_k$$

- FedAvg:  $w_k = 1/K$ .
- WFedAvg:  $w_k = |\mathcal{D}_k| / (\sum_{j=1}^K |\mathcal{D}_j|)$ .

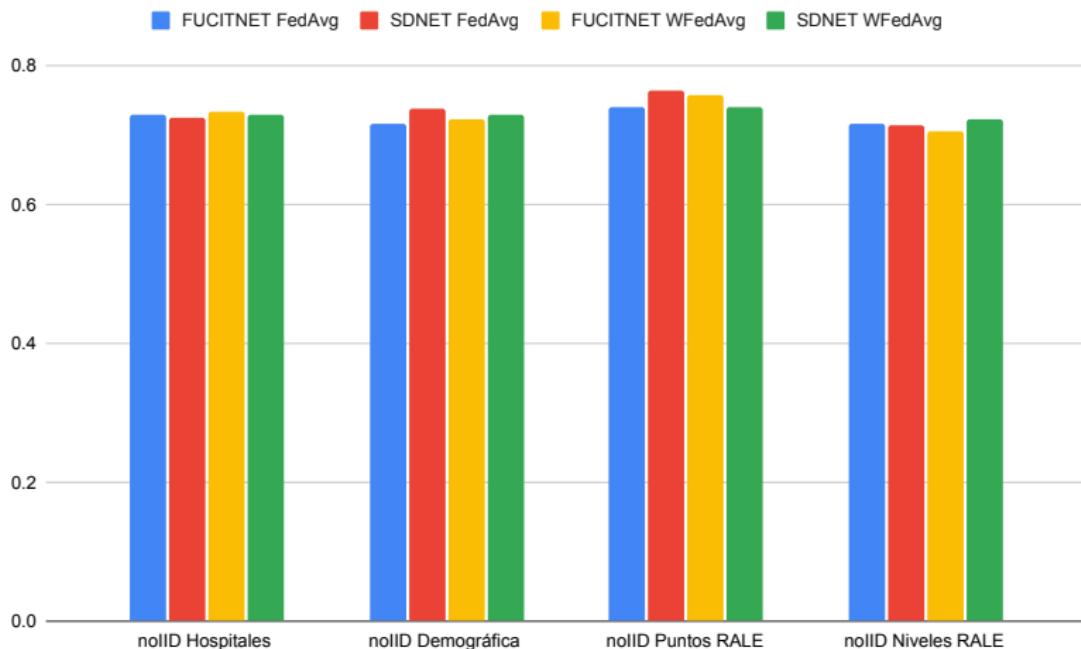
## Configuraciones

- **Configuración IID:** los datos se reparten idénticamente e independientemente entre los nodos prefijados. Consideramos 4 particiones: 1, 3, 6 y 9 nodos.
- **Configuración noIID:** cada nodo tiene una distribución distinta. Repartimos los datos de acuerdo a 4 criterios, obteniendo las particiones: Hospitales, Demográfica, Puntos RALE, Niveles RALE.

## Análisis de resultados: resumen accuracy IID



## Análisis de resultados: resumen accuracy nIID



## Análisis de resultados: resumen accuracy global

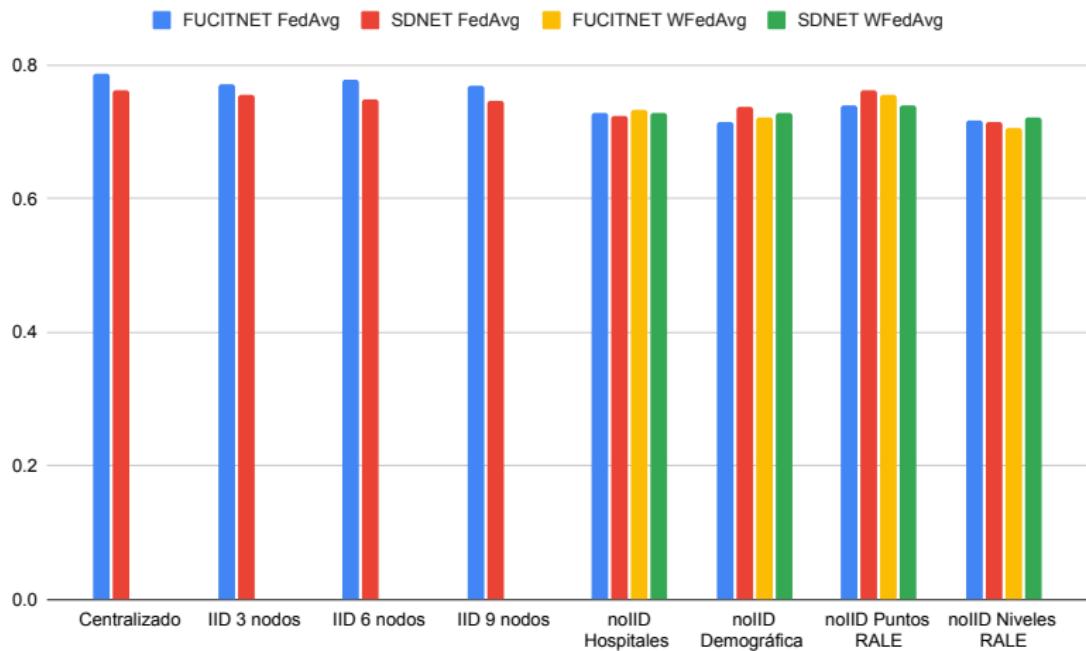
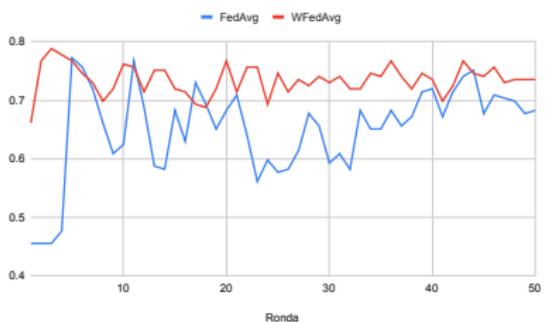
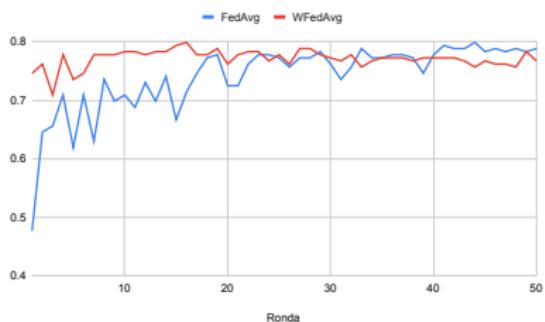


Figura: Resultados obtenidos en cada escenario y para cada modelo expresados en términos de accuracy.

# Convergencia: FedAvg vs WFedAvg



(a) FUCITNET.



(b) SDNET.

## Convergencia: ¿cuándo parar?

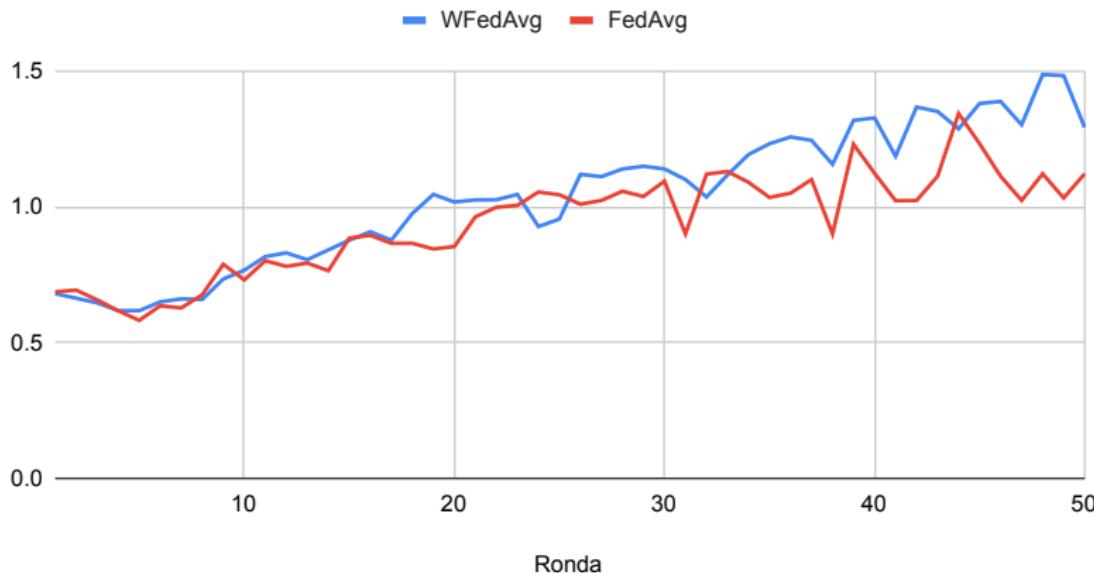


Figura: Evolución de la función de pérdida de FUCITNET en una de las ejecuciones de entrenamiento.

- 1 Introducción
  - 2 Matemáticas
  - 3 Informática teórica
  - 4 Informática práctica
  - 5 Conclusiones y vías futuras

## Conclusiones

- Comprensión profunda del problema desde el punto de vista teórico.
  - Implementación software estable y consistente.
  - Resultados coherentes en el escenario centralizado con el artículo original de COVID-SDNET.
  - Escenario federado: extensión de la metodología es robusta y estable. Aproximación satisfactoria de un problema real → ¡Resultados prometedores!

## Vías futuras

- Ampliar el conjunto de datos COVIDGR-FL → hospitales bien representados.
  - Plantear nuevos operadores de agregación.
  - Proponer mecanismos para detectar la convergencia del modelo y evitar sobreajuste.
  - Plano teórico: formalismo de aprendizaje federado.

**¡Gracias por su atención!**