Reconnaissance vocale lors d'appel d'urgence grâce à un réseau de neurones

TRAN-THUONG Tien-Thinh [n°30903]

2021-2022

Sommaire

Sommaire

Sommaire

Sommaire

Un cas de non convergence de la descente de gradient

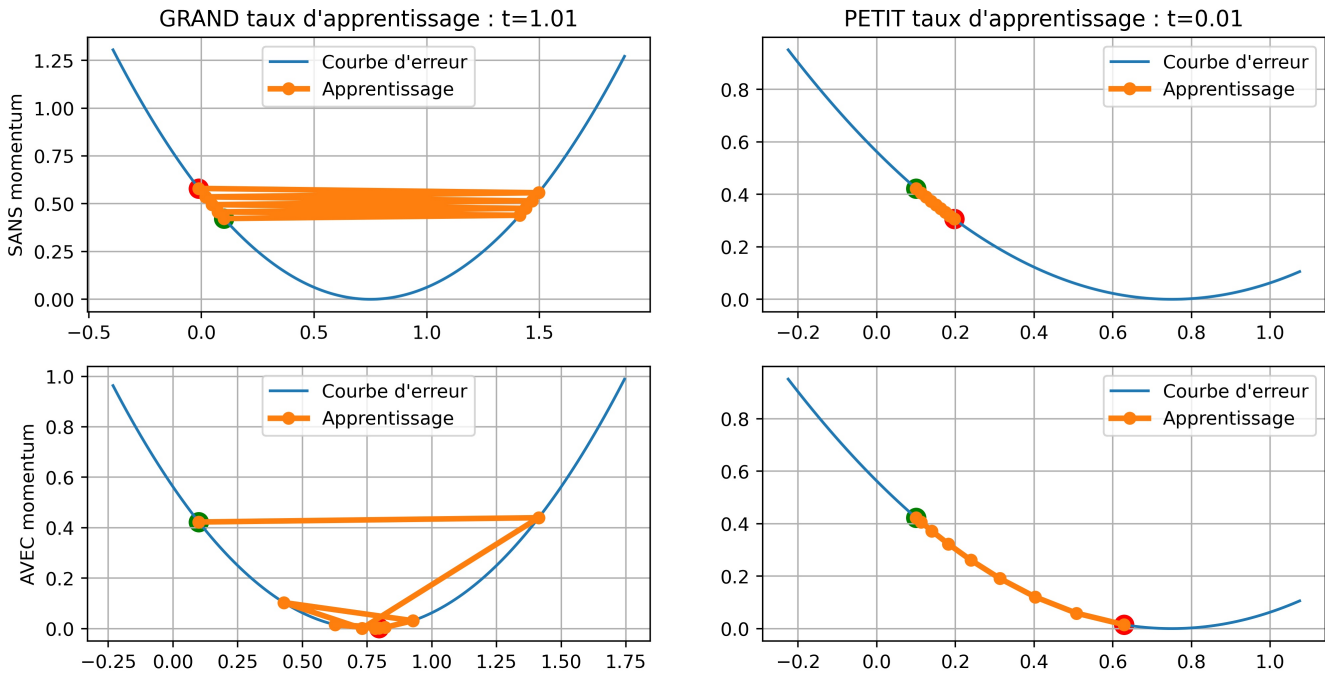


Figure – Descente de gradient SANS/AVEC momentum où $\gamma = 0.5$

La terminaison de la descente de gradient

En pratique

Pour terminer les itérations de la rétropropagation sur un réseau de neurones trois conditions de terminaison existent :

- Un nombre maximal d'itération de la rétropropagation
- Un seuil minimal pour l'erreur
- Un seuil minimal pour la norme infinie du gradient

Ainsi, en pratique, si une de ces trois conditions est atteinte, l'algorithme se termine. Le variant de boucle du nombre d'itération assure alors la terminaison de l'algorithme

Les hypothèses pour la terminaison

En théorie

Une fonction de R^n dans R de classe C^2, strictement convexe et coercive ($\lim_{\|x\|\to+\infty} f(x) = +\infty$) admet un unique minimum. On peut donc trouver une suite t_k de taux d'apprentissage de sorte que la descente de gradient converge.

Exemple avec les hypothèses

La relation de récurrence s'écrit alors :

$$x_{k+1} = x_k - t_k \nabla f(x_k)$$

avec x_k tendant vers l'unique minimum de f

La rétropropagation sur le XOR

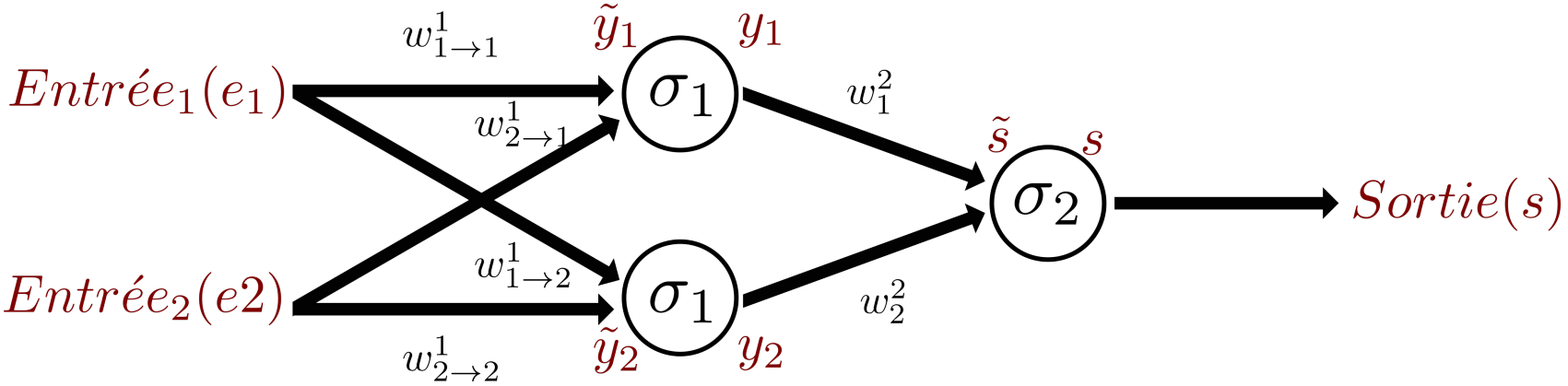


Figure – Schéma du réseau de neurone reproduisant le XOR

$$\begin{cases} \tilde{y}_1 = w^1_{11} \times e_1 + w^1_{21} \times e_2 \\ \tilde{y}_2 = w^1_{12} \times e_1 + w^1_{22} \times e_2 \\ \tilde{s} = w^2_1 \times y_1 + w^1_{11} \times y_2 \end{cases} \quad et \quad \begin{cases} y_1 = \sigma_1(\tilde{y}_1) \\ y_2 = \sigma_1(\tilde{y}_2) \\ s = \sigma_2(\tilde{s}) \end{cases} \tag{1}$$

Une simplification matricielle

Convention adoptée

- @ Le produit matricielle
- $*$ Le produit d'Hadamard
- f La fonction d'erreur $f : s \to (s - s_{attendue})^2$

$$\begin{cases} \begin{pmatrix} \tilde{y}_1 \\ \tilde{y}_2 \end{pmatrix} = \begin{pmatrix} w_{11}^1 & w_{21}^1 \\ w_{12}^1 & w_{22}^1 \end{pmatrix} @ \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} \\ \begin{pmatrix} \tilde{s} \end{pmatrix} = \begin{pmatrix} w_1^2 & w_2^2 \end{pmatrix} @ \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \end{cases} \tag{2}$$

La rétropropagation matricielle

$$\begin{cases} \left(\frac{\partial f}{\partial w_1^2}(w_1^2) \quad \frac{\partial f}{\partial w_2^2}(w_1^2)\right) = \left(\frac{\partial f}{\partial s}(s)\right) * \left(\sigma_2'(\tilde{s})\right) @ \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}^t \\ \begin{pmatrix} \frac{\partial f}{\partial w_{11}^2}(w_{11}^2) & \frac{\partial f}{\partial w_{21}^2}(w_{21}^2) \\ \frac{\partial f}{\partial w_{12}^2}(w_{12}^2) & \frac{\partial f}{\partial w_{22}^2}(w_{22}^2) \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial y_1}(y_1) \\ \frac{\partial f}{\partial y_2}(y_2) \end{pmatrix} * \begin{pmatrix} \sigma_1'(\tilde{y}_1) \\ \sigma_1'(\tilde{y}_2) \end{pmatrix} @ \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}^t \end{cases} \tag{3}$$

avec

$$\left\{ \begin{pmatrix} \frac{\partial f}{\partial y_1}(y_1) \\ \frac{\partial f}{\partial y_2}(y_2) \end{pmatrix}^t = \left(\frac{\partial f}{\partial s}(s)\right) * \left(\sigma_2'(\tilde{s})\right) @ \begin{pmatrix} w_1^2 & w_2^2 \end{pmatrix} \right. \tag{4}$$

Des fonctions d'activation

Fonction	Formule	Dérivée
Sigmoïde	$\frac{1}{1+e^{-x}}$	$f(x) \times (1 - f(x))$
Tangente Hyperbolique (Tanh)	$\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	$1 - f(x)^2$
Unité Linéaire Rectifiée (ReLU)	$max(0, x)$	$\begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{sinon} \end{cases}$

La fonction d'activation : Sigmoïde

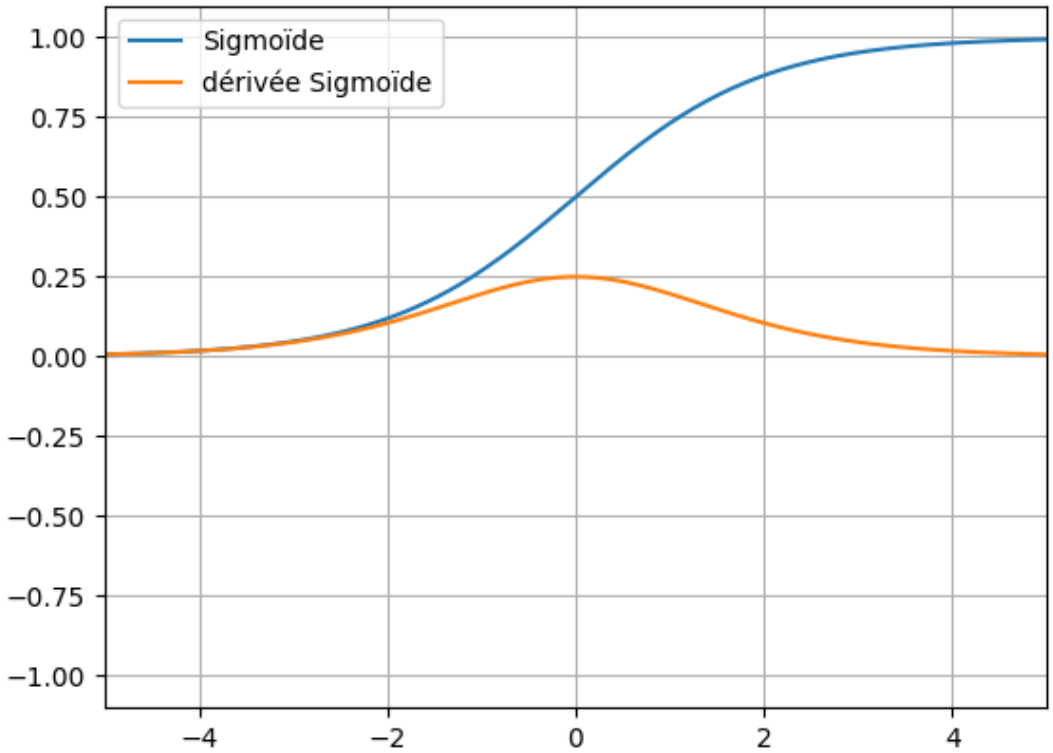


Figure – Sigmoïde

La fonction d'activation : TANH

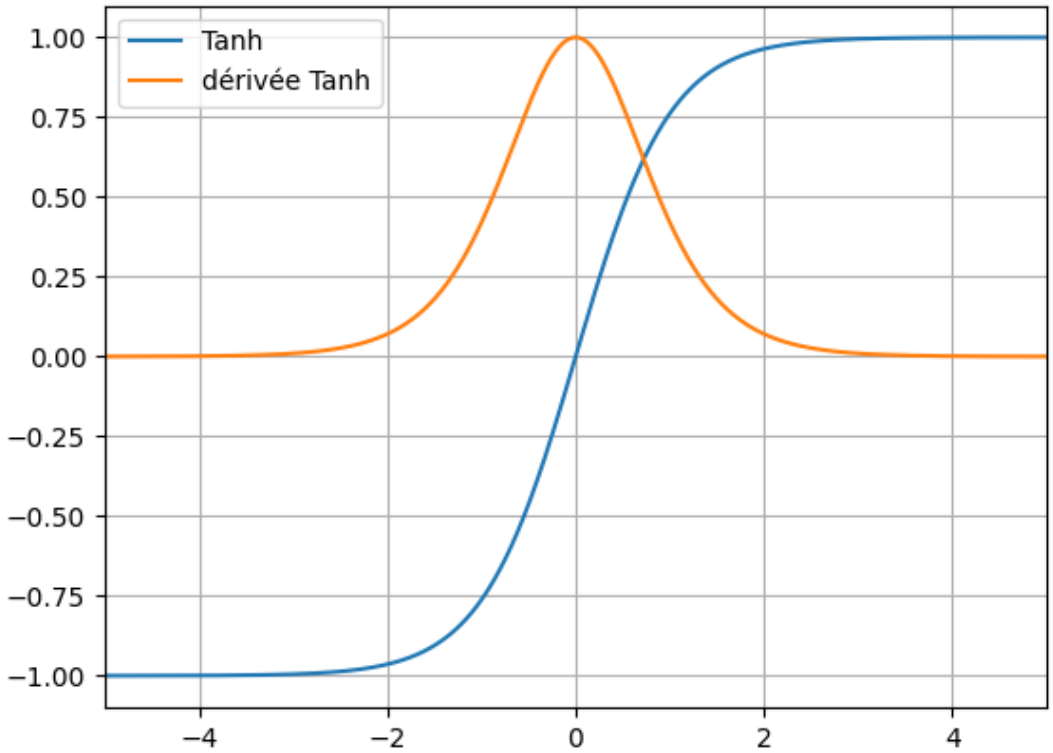


Figure – TANH

La fonction d'activation : ReLu

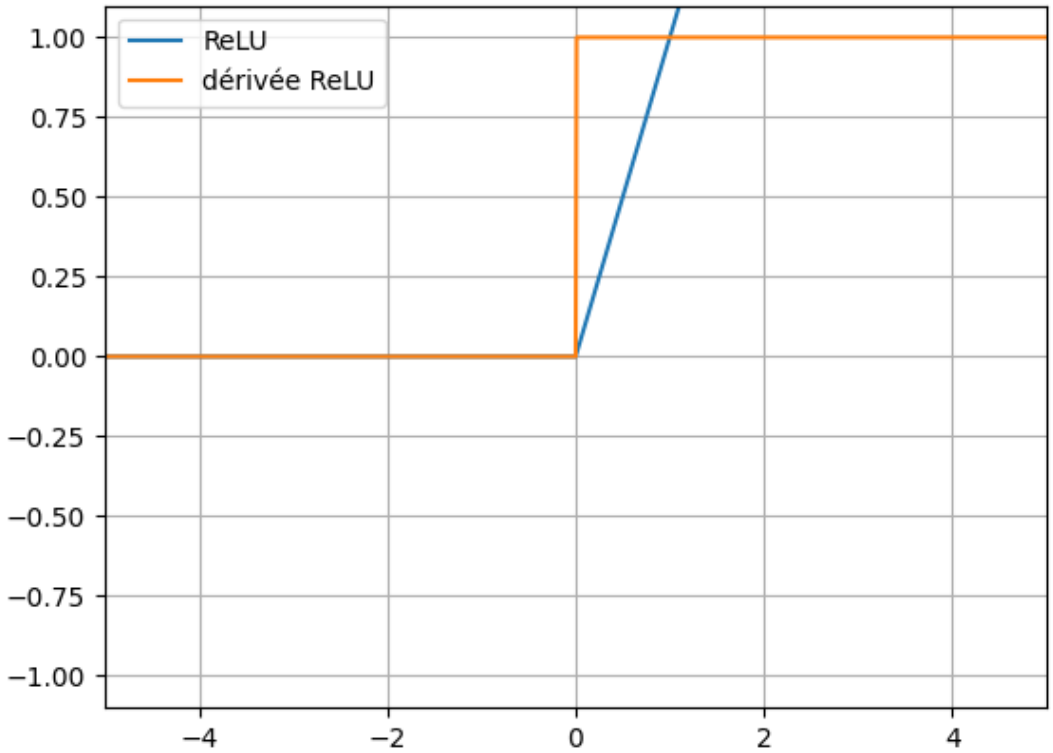


Figure – ReLu

La rétropropagation pour la classification

Cross-entropy

La fonction d'erreur des problèmes de classification est Cross-entropy :

- $L = -\sum_{k=1}^{n} y_i log(p_i)$ avec y_i la sortie attendue
- $\frac{\partial L}{\partial a_i} = p_i - y_i$

Une base de données plus complexe

Description

Images de taille 28×28 pixels en noir et blanc :
- 60 000 images pour l'entrainement.
- 10 000 autres pour la vérification.

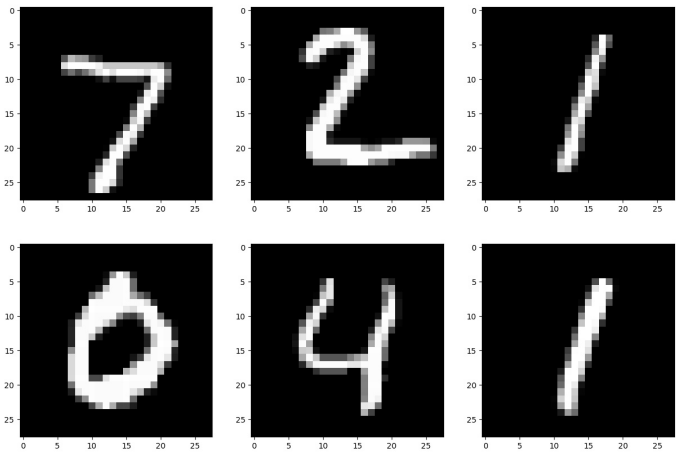


Figure – MNIST

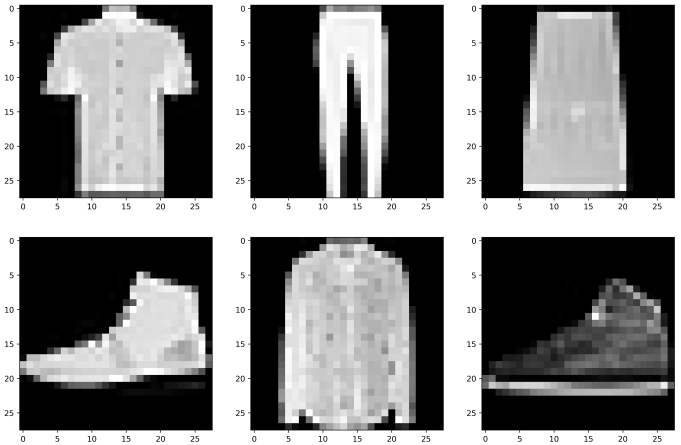
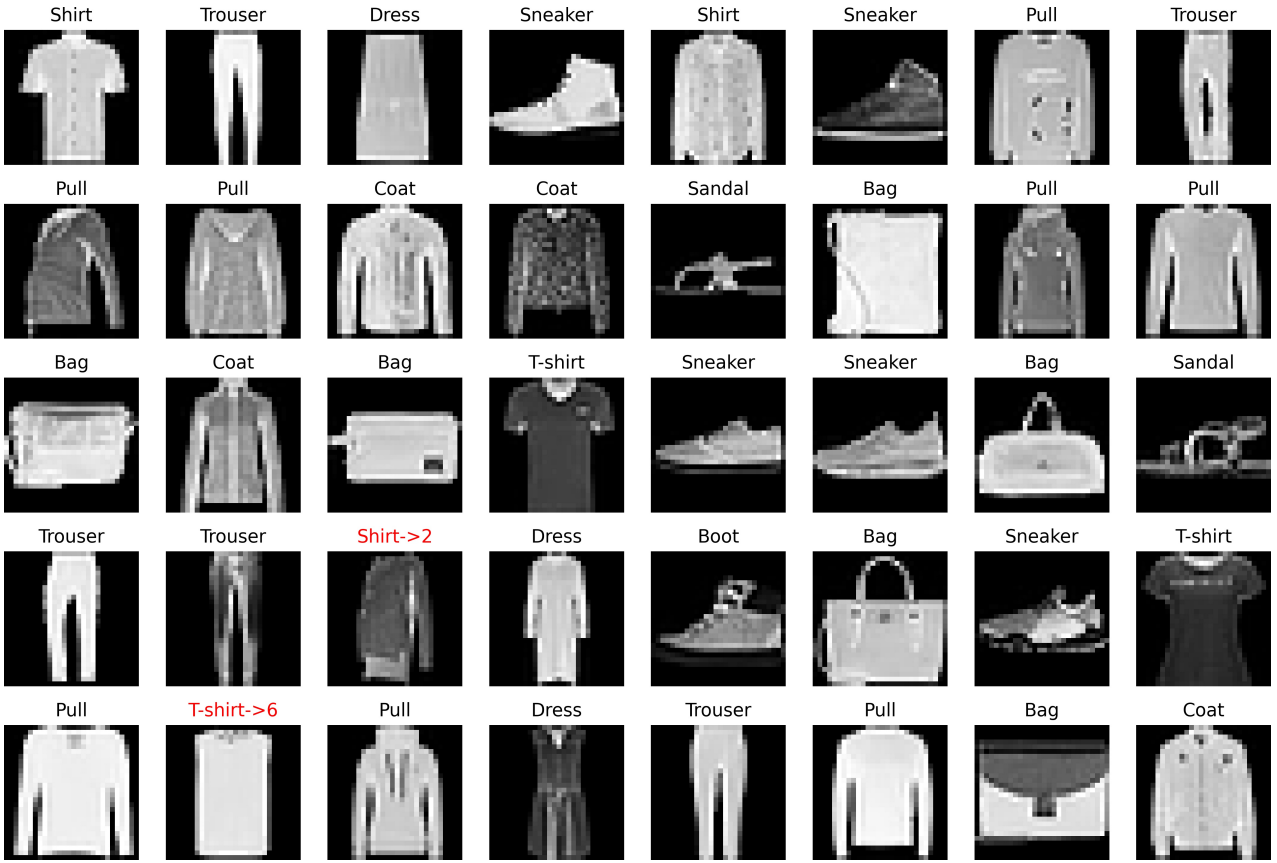


Figure – Fashion MNIST

Mes résultats




```
dico = {
      0: 'T-shirt',
      1: 'Trouser',
      2: 'Pull',
      3: 'Dress',
      4: 'Coat',
      5: 'Sandal',
      6: 'Shirt',
      7: 'Sneaker',
      8: 'Bag',
      9: 'Boot'
}
```

Figure – Exemple sur un échantillon de 40 images *Fashion MNIST*

Réseau de neurones modèle

```
 Layer (type)                 Output Shape              Param #
=================================================================
 resizing (Resizing)          (None, 32, 32, 1)         0
 conv2d (Conv2D)              (None, 30, 30, 32)        320
 conv2d_1 (Conv2D)            (None, 28, 28, 64)        18496
 max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
 dropout (Dropout)            (None, 14, 14, 64)        0
 flatten (Flatten)            (None, 12544)             0
 dense (Dense)                (None, 128)               1605760
 dropout_1 (Dropout)          (None, 128)               0
 dense_1 (Dense)              (None, 8)                 1032
=================================================================
Total params: 1,625,608
Trainable params: 1,625,608
Non-trainable params: 0
```

Réseau de neurones adapté

```
 Layer (type)                 Output Shape              Param #
=================================================================
 resizing (Resizing)          (None, 32, 32, 1)         0
 conv2d (Conv2D)              (None, 30, 30, 32)        320
 conv2d_1 (Conv2D)            (None, 28, 28, 64)        18496
 max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
 dropout (Dropout)            (None, 14, 14, 64)        0
 flatten (Flatten)            (None, 12544)             0
 dense (Dense)                (None, 128)               1605760
 dropout_1 (Dropout)          (None, 128)               0
 adaptation (Dense)           (None, 12)                1548
=================================================================
Total params: 1,626,124
Trainable params: 1,548
Non-trainable params: 1,624,576
```