Minería de Datos: Aspectos Avanzados

Usos del suelo desde el espacio

Índice

Competición: Usos del suelo desde el espacio

Se trata de un problema de clasificación cuyo objetivo consiste en identificar **29 clases de suelos diferentes** a partir de imágenes satelitales de 224x224 píxeles y 3 canales de colores (RGB).

El **conjunto de entrenamiento** dispone de un total de **354 imágenes para cada clase**, mientras que el **conjunto de test contiene 1.618 imágenes** considerando todas las categorías disponibles.

Enfoques para resolver el problema

1. ***Transfer Learning.***
   a. Utilizar modelos pre-entrenados sobre *ImageNet* para utilizar su conocimiento y construir clasificadores de manera directa.
   b. Utilizar las arquitecturas definidas de los modelos para entrenarlos parcial o totalmente de modo que podamos adaptarlos lo máximo posible al problema en cuestión.

2. ***Data Augmentation.***
   a. Generar nuevas imágenes diferentes a las originales para intentar obtener una mayor robustez al entrenar los clasificadores.
   b. Aumentar el número de ejemplares de cada clase para disponer de una mayor representatividad que intente mejorar la capacidad de predicción de los modelos.

3. ***AutoML.*** Buscar de forma automática la arquitectura y su configuración asociada más apropiada para el problema de clasificación. Solo depende de los datos de entrenamiento minimizando la intervención humana.

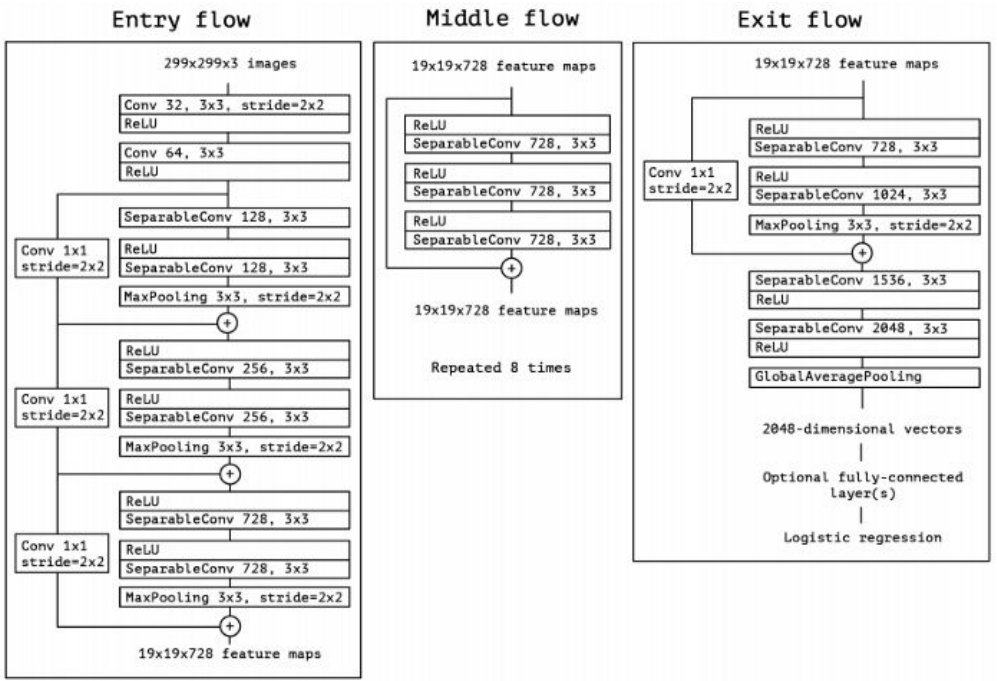
Experimentación de Lidia Sánchez
Usuario en Kaggle: Lidia

Lectura y preprocesamiento de las imágenes

- ***ImageDataGenerator***
  - Lee las imágenes de la jerarquía de carpetas original para transformarlas a matrices que se pueden utilizar directamente.
  - Escala los valores de las matrices en un rango de valores entre 0 y 1.
  - Reserva un 20% del conjunto de entrenamiento para realizar validación tras cada época.
- ***Data Augmentation*** con *ImageDataGenerator.*
  - Aplica las transformaciones especificadas sobre las imágenes del conjunto de entrenamiento sin aumentar su tamaño.
  - Mejor flujo de transformaciones encontrado:
    - Rotación 45º.
    - Recorte vertical y horizontal de 0.15.
    - Modo espejo vertical y horizontal.
    - Zoom del 50%.
- ***Data Augmentation*** con *ImageDataGenerator* + Conjunto de entrenamiento original.
  - Genera 5 nuevas imágenes aplicando las transformaciones por cada imagen de entrenamiento.
  - Almacena las imágenes transformadas junto al conjunto de datos de entrenamiento.

Modelo Xception: arquitectura

- **132 capas.**
- ***Separable convolutions***. Aplica convolución sobre cada canal de la imagen y luego combina los resultados.
  - Menor número de recursos.
  - Mejora el rendimiento del modelo.



Modelo Xception: configuraciones de entrenamiento

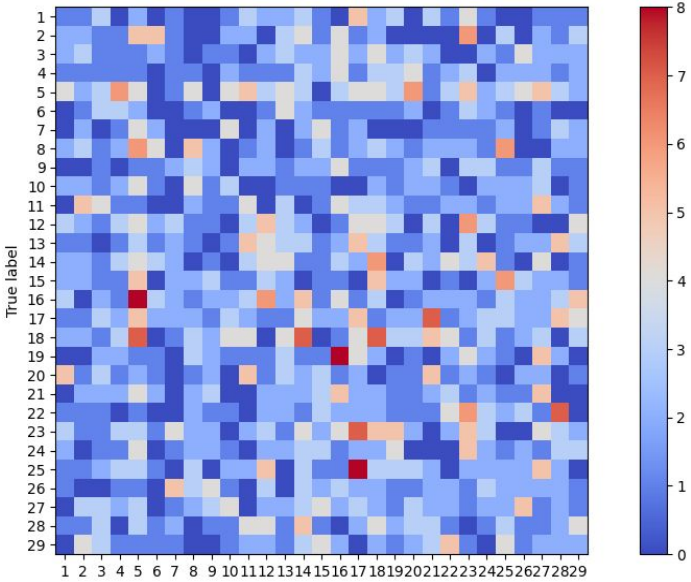
- **Uso del modelo pre-entrenado**.
- **Re-entrenamiento del modelo.**
  - A mayor número de capas re-entrenadas, menor *accuracy*.
  - Entrenando el modelo al completo con una tasa de aprendizaje menor (1e4).

25 épocas
128 iteraciones/época

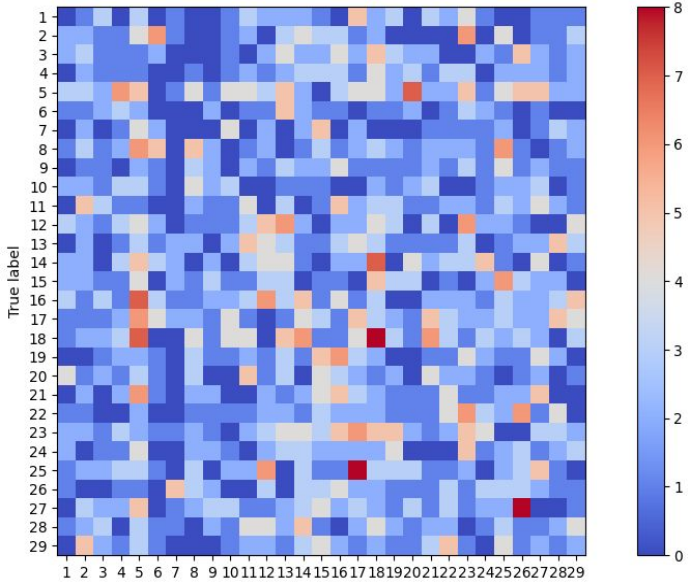
Uso del modelo pre-entrenado.	72.86%
Re-entrenando a partir de la capa nº 100.	**93.81%**
Re-entrenando a partir de la capa nº 100 y tasa de aprendizaje de 1e4.	92.47%
Re-entrenando a partir de la capa nº 50.	89.48%
Re-entrenando a partir de la capa nº 10.	84.94%
Entrenando el modelo completo con una tasa de aprendizaje de 1e4.	91.85%

Modelo Xception: experimentos con *Data Augmentation*

- **Disminuye la métrica *accuracy* si no se entrena el modelo al completo.**
- Necesita una **tasa de aprendizaje menor** (1e4) para mejorar la predicción.



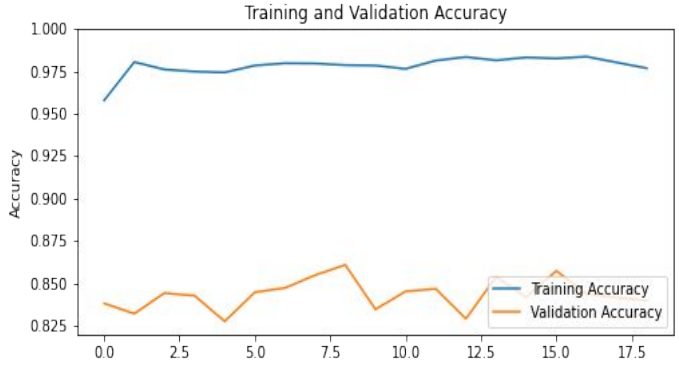
***Data Augmentation - accuracy:* 94.63%**



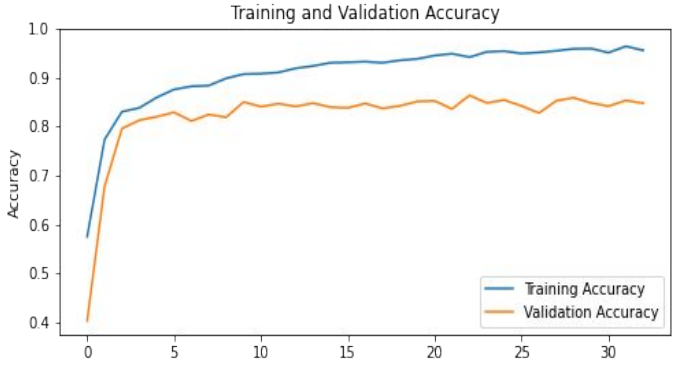
***Data Augmentation* + Conjunto original - *accuracy:* 92.16%**

Modelo Xception: técnicas de regularización

- ***Early Stopping***. Ayuda a evitar el sobreajuste del modelo y ahorra tiempo.
  - Parar tras 10 iteraciones sin mejorar la métrica *val_accuracy* (*accuracy* sobre el conjunto de validación).
- ***L1.*** Utiliza el valor absoluto de los pesos como estrategia de penalización.
  - Genera pesos iguales a 0 y decrementa el valor de *accuracy*.
- ***L2 (weight decay)***. Utiliza los pesos al cuadrado para aplicar la penalización.
  - **Incrementando el factor de penalización se reduce el sobreajuste y aumenta el *accuracy*.**
- **Combinando L1 y L2** el aprendizaje del modelo es más estable.



Regularización L2 - *accuracy*: 95.15%



Combinación de L1 y L2 - *accuracy:* 94.43%

Modelo Xception: experimentos con optimizadores

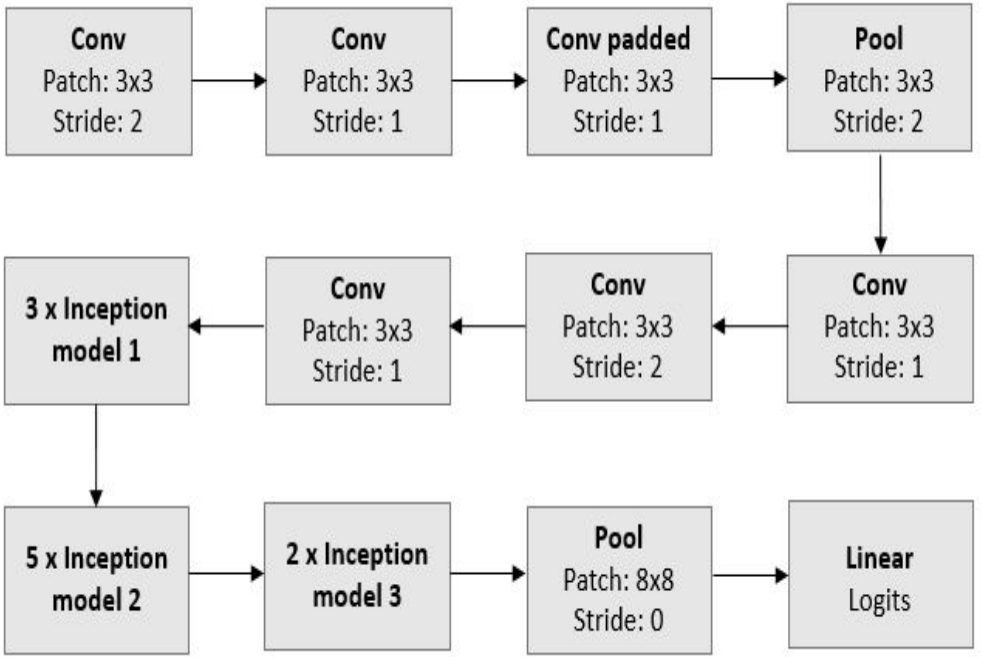
Configuración común de entrenamiento:

- Entrenamiento del modelo al completo
- Data Augmentation
- Tasa de aprendizaje = 1e4
- Early Stopping
- Regularización L2(0.1)

Adam	Actualiza los pesos de las neuronas localmente. Es más eficiente computacional y temporalmente, además de proveer mejores resultados.	Accuracy: 95.15%
Adamax	Basado en el optimizador *Adam* utiliza otra norma para actualizar los pesos que puede ayudar a clasificar datos con *outliers*. Más costoso temporalmente y con menor *accuracy*.	Accuracy: 92.89%
Nadam	Basado en el optimizador *Adam* calcula una estimación de la siguiente posición del gradiente. Más eficiente temporalmente pero con un rendimiento similar a Adam basado en *accuracy*.	Accuracy: 94.22%

Modelo InceptionV3: arquitectura

- **311 capas.**
- **Capas convolutivas con filtros de menor dimensión** para agilizar el cálculo de pesos.
- **Capas convolutivas asimétricas** para mejorar costes computacionales.
- **Clasificadores auxiliares** que actúan como regularizadores.



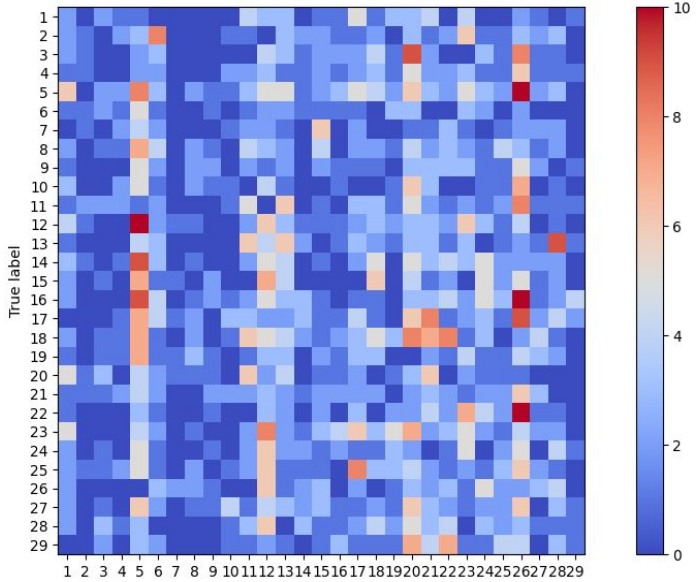
Modelo InceptionV3: configuraciones de entrenamiento

- **No se hace uso del modelo pre-entrenado** porque ya conocemos que no proporciona buenos resultados según hemos podido comprobar con *Xception.*
- **Re-entrenamiento del modelo.**
  - Menor *accuracy* entrenando solo algunas capas que el modelo completo.
  - Entrenando el modelo completo con una tasa de aprendizaje mucho menor (1e5) al ser una red más profunda, aunque no necesita más épocas.

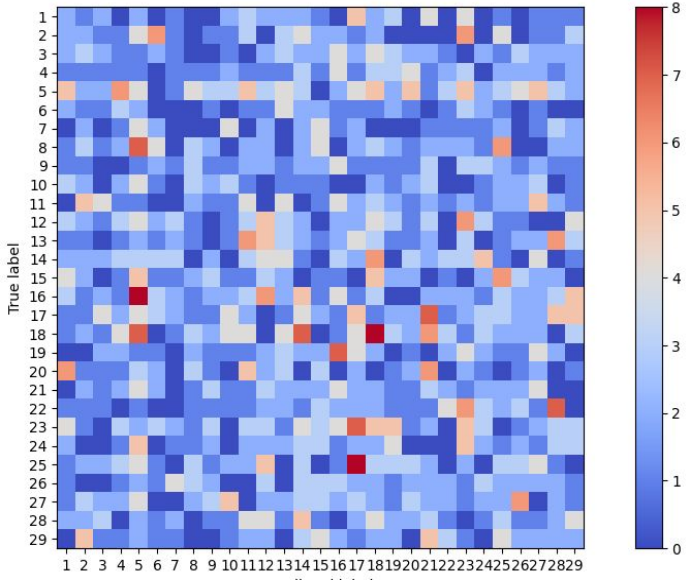
Re-entrenando a partir de la capa nº 100 con 25 épocas y 128 iteraciones/época.	77.42%
Entrenando el modelo completo con tasa de aprendizaje=1e4 con 25 épocas y 128 iteraciones/época.	88.65%
Entrenando el modelo completo con tasa de aprendizaje=1e5 con 25 épocas y 128 iteraciones/época.	**95.46%**
Entrenando el modelo completo con tasa de aprendizaje=1e5 con 35 épocas y 128 iteraciones/época.	**94.53%**

Modelo InceptionV3: experimentos con *Data Augmentation*

- **Inviable entrenar un modelo con el conjunto original combinado con el de *Data Augmentation*** por la gran cantidad de recursos y tiempo necesarios.
- Fundamental añadir ***Early Stopping*** para evitar que el modelo sobreajuste.



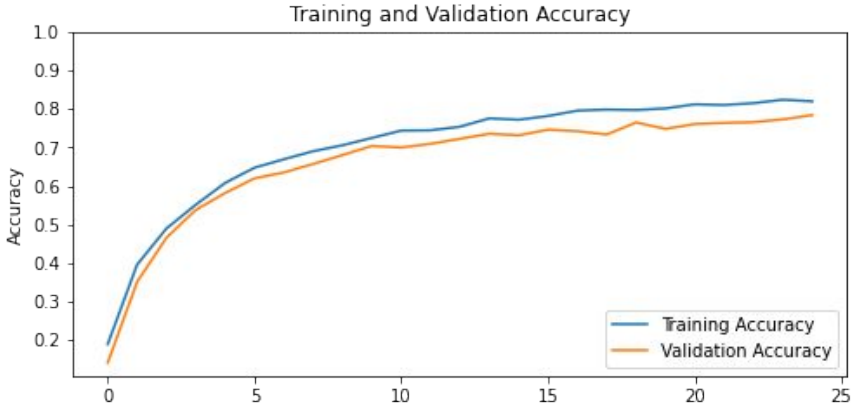
***Data Augmentation* - *accuracy:* 53.60%**



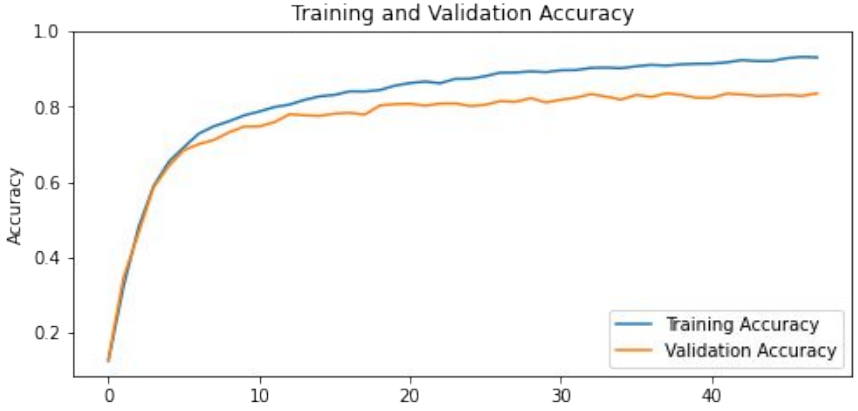
***Data Augmentation* + Early Stopping - *accuracy:* 93.26%**

Modelo InceptionV3: técnicas de regularización

- ***Early Stopping***. Fundamental para evitar que el modelo sobreajuste.
  - Parar tras 10 iteraciones sin mejorar la métrica *val_accuracy* (*accuracy* sobre el conjunto de validación).
- ***L1.*** Utiliza el valor absoluto de los pesos como estrategia de penalización.
  - Genera pesos iguales a 0 y decrementa el valor de *accuracy*.
- ***L2 (weight decay)***. Utiliza los pesos al cuadrado para aplicar la penalización.
  - Penaliza demasiado durante el aprendizaje y disminuye el valor de *accuracy*.
- **Combinando L1 y L2 el aprendizaje es similar pero aumenta el *accuracy* evitando el sobrejauste.**



Regularización L2 - *accuracy*: 76.52%



Combinación de L1 y L2 - *accuracy:* 94.43%

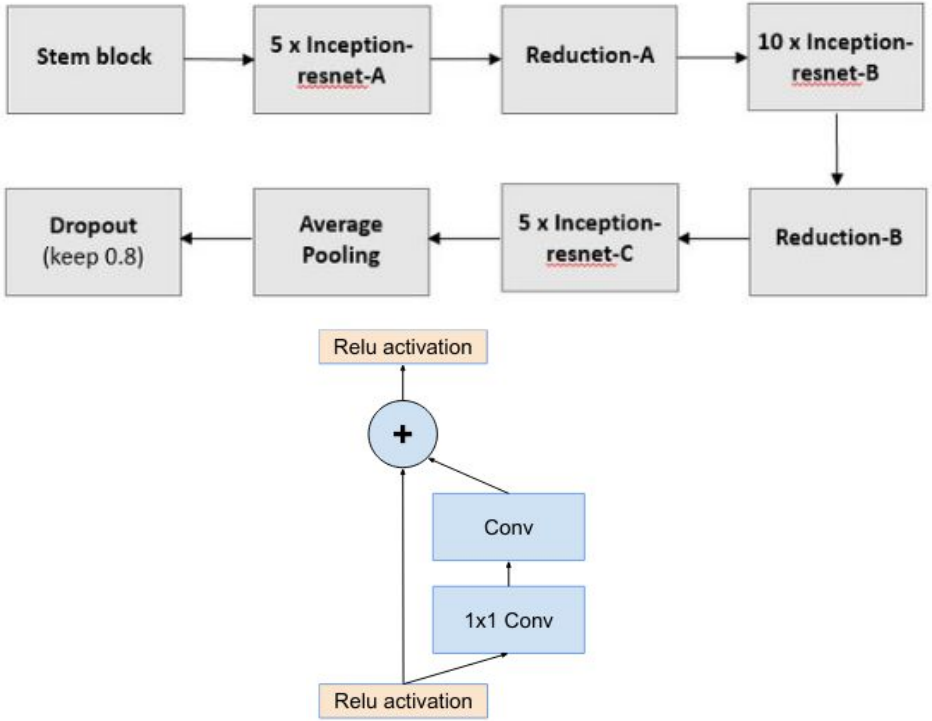
Modelo InceptionV3: experimentos con optimizadores

Configuración común de entrenamiento:
- Entrenamiento del modelo al completo
- Conjunto de entrenamiento original
- Tasa de aprendizaje = 1e5
- Early Stopping

Adam	Actualiza los pesos de las neuronas localmente. Es más eficiente computacional y temporalmente, además de proveer mejores resultados.	Accuracy: 95.46%
Adamax	Basado en el optimizador *Adam* utiliza otra norma para actualizar los pesos que puede ayudar a clasificar datos con *outliers*. **Inviable por recursos computacionales y temporales.**	-
Nadam	Basado en el optimizador *Adam* calcula una estimación de la siguiente posición del gradiente. Más eficiente temporalmente pero peores resultados en *accuracy*.	Accuracy: 88.88%

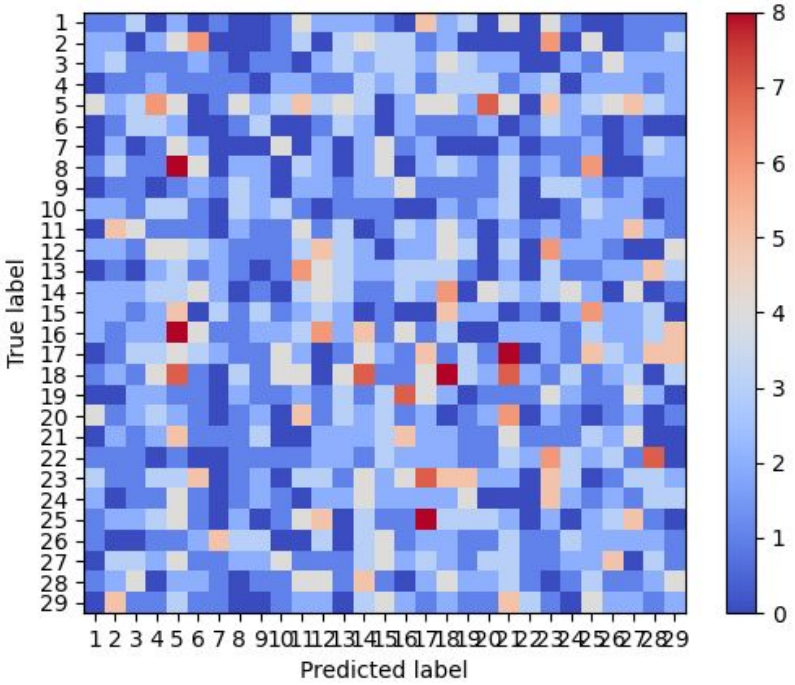
Modelo InceptionResnetV2: arquitectura

- **780 capas.**
- **Arquitectura base: Inception.**
  - **Varias subredes** más sencillas para entrenarlas en paralelo.
- **Aprendizaje residual:** añade nuevas entradas a la red para generar nuevas salidas y evitar la degradación en redes muy profundas.
- ***Batch-normalization*** solo sobre capas de menor tamaño para evitar el consumo desorbitado de recursos.



Modelo InceptionResnetV2: configuraciones de entrenamiento

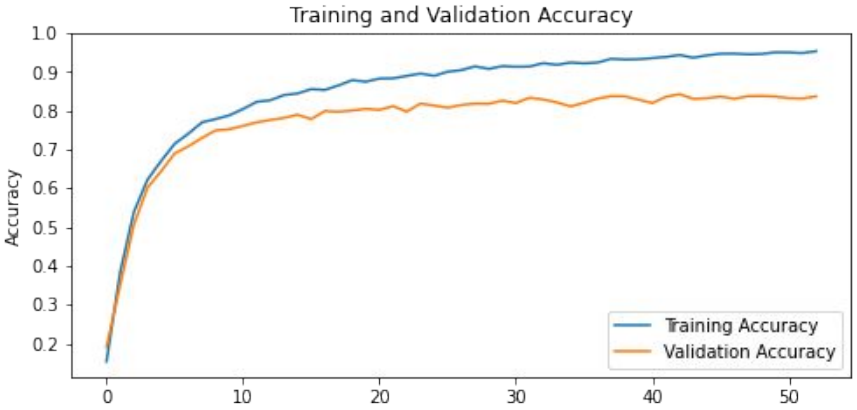
- En base a los experimentos realizados anteriormente, la construcción de modelos con esta arquitectura es la siguiente:
  - **Entrenamiento del modelo completo.**
  - ***Data Augmentation*.**
  - **Tasa de aprendizaje=1e5.**
  - **Early Stopping.**



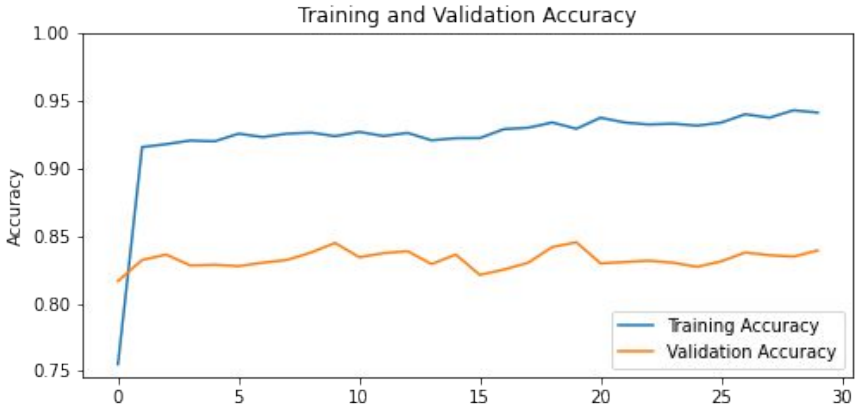
***Accuracy:* 92.37%**

Modelo InceptionResnetV2: técnicas de regularización

- ***Early Stopping***. Fundamental para evitar que el modelo sobreajuste.
  - Parar tras 10 iteraciones sin mejorar la métrica *val_accuracy* (*accuracy* sobre el conjunto de validación).
- ***L1.*** Utiliza el valor absoluto de los pesos como estrategia de penalización.
  - Genera pesos iguales a 0 y decrementa el valor de *accuracy*.
- ***L2 (weight decay)***. Utiliza los pesos al cuadrado para aplicar la penalización.
  - **Incrementando el factor de penalización se reduce el sobreajuste y aumenta el *accuracy*.**
- **Combinando L1 y L2** el aprendizaje crece rápidamente pero es más irregular y menos efectivo.



Regularización L2 - *accuracy*: 93.60%



Combinación de L1 y L2 - *accuracy:* 91.66%

Ensembles

- **Objetivo:** combinar los mejores modelos obtenidos para aumentar la capacidad de predicción.
- Diferentes arquitecturas, configuración y conjuntos de entrenamiento.
- **Votación mayoritaria** a partir de varios modelos.
  - La clase más votada.
- **Votación ponderada** a partir de varios modelos.
  - Mayor peso a los que disponen de mayor *accuracy* para darle mayor peso a "los mejores".
  - La clase más votada.
  - **No se han observado mejoras** en el *accuracy* con respecto a la votación mayoritaria.

1. **Clase mayoritaria: 5 modelos Xception**
   a. ***Accuracy:* 97.21%**
   b. *Accuracy* de los modelos: 92% - 94%
2. **Clase mayoritaria: Xception + InceptionV3**
   a. ***Accuracy: 96.49%***
   b. Modelos Xception anteriores.
   c. 2 modelos InceptionV3: 94.53% y 95.46%
3. **Clase mayoritaria: Xception + InceptionV3**
   a. ***Accuracy: 97.62%***
   b. 6 modelos Xception: 92% - 94%
   c. 2 modelos InceptionV3: 94.53% y 95.46%
4. **Votación ponderada: Xception+InceptionV3**
   a. ***Accuracy: 97.42%***
   b. Misma combinación de modelos anterior
   c. **Minimiza la aportación de los modelos con menor *accuracy*** y no permite corregir muchas de las clases mal clasificadas por los mejores modelos.

Mejores modelos obtenidos

Según la validación sobre el 60% de test

Ensemble de **9 clasificadores** combinando modelos **Xception e InceptionV3 con un rango de *accuracy* entre 92% y 95%** utilizando la votación de la clase mayoritaria sin ponderación.

Accuracy sobre el 60%: 97.93%
Accuracy sobre el 100%: 97.22%

Según la validación sobre el 100% de test

Ensemble de **8 clasificadores** combinando modelos **Xception e InceptionV3 con un rango de *accuracy* entre 92% y 95%** utilizando la votación de la clase mayoritaria sin ponderación.

Accuracy sobre el 60%: 97.06%
Accuracy sobre el 100%: 97.42%

En ambos modelos se ha podido identificar una **mayor concentración de errores** en los siguientes tipos de suelos:

- Gran dificultad al identificar el tipo de suelo: *5_SrublandClose*
- Confusión generalizada al identificar tres **tipos de bosques**:
  - *16_ForestsClEvNe*
  - *17_ForestsDeEvNe*
  - *18_WetlandMangro*

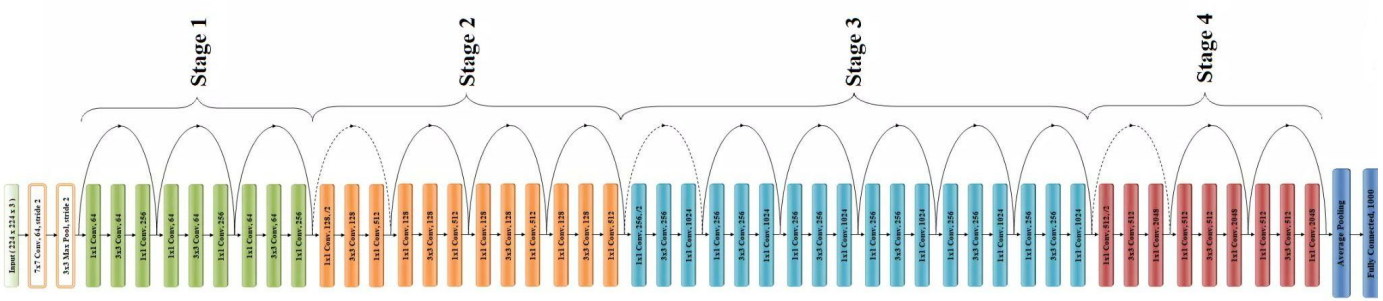
Experimentación Javier Martínez Portellano
Usuario en kaggle: nombre completo

Lectura y preprocesamiento de las imágenes

- ***ImageDataGenerator***
  - Lee las imágenes de la jerarquía de carpetas original para transformarlas a matrices que se pueden utilizar directamente.
  - Escala los valores de las matrices en un rango de valores entre 0 y 1.
  - Reserva un 20% del conjunto de entrenamiento para realizar validación tras cada época.
- ***Maximización de resultados con el conjunto base de imágenes***
- ***Búsqueda de hiperparametros mediante fine-tuning***

Modelo ResNet50

- Compuesto por 50 capas distribuidas en 4 fases
- Uso de residuos para reducir el error producido durante el entrenamiento
- Aumento de la facilidad de optimización
- Uso de diferentes configuraciones para el tamaño de las capas convolucionales



ResNet50: configuración de entrenamiento

- Uso de diferentes configuraciones en el número de capas entrenadas.
- Estudio del efecto de las épocas usadas
- Modificación del ratio de aprendizaje

Para el resto de parámetros:

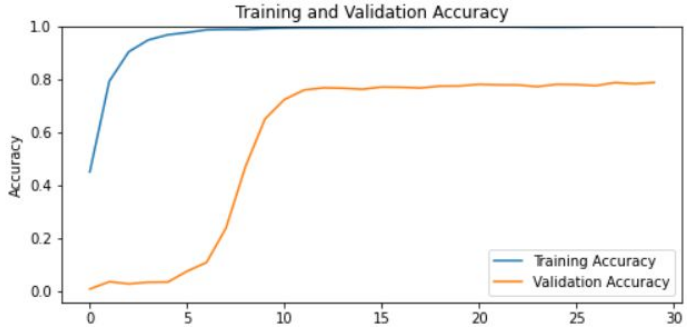
- Optimizador adam
- Activación softMax
- Tamaño de batch: 64

ResNet50: resultados

Epochs	Ratio de aprendizaje	Capas entrenadas	Accuracy
25	0.001	50	0.51
25	0.001	100	0.76
25	0.001	ALL	0.94
30	0.001	ALL	0.84
30	0.0001	ALL	0.91
25	0.0001	ALL	0.92

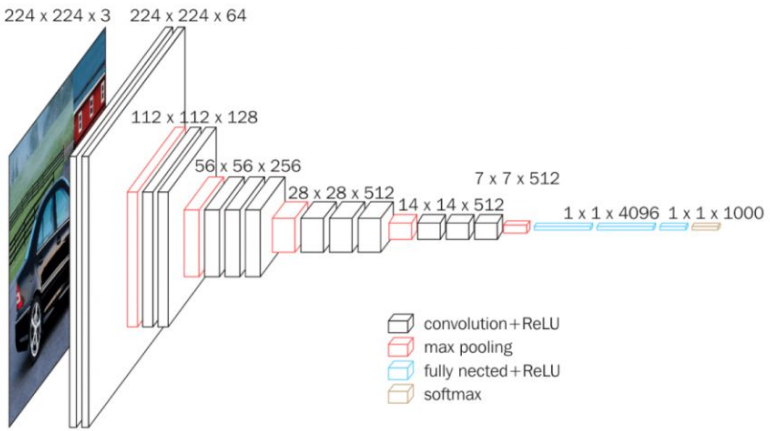
ResNet50: conclusiones

- Para los modelos con mejor calidad se han entrenado como mínimo dos modelos para estudiar la variabilidad entre modelos con los mismos hiperparametros
- Una tasa de aprendizaje menor aumenta la calidad del modelo si se acompaña de un entrenamiento más prolongado
- Es necesario reentrenar todas las capas para obtener un mejor modelo



Modelo VGG16

- Uso de nucleos de 3x3 uno tras otro a diferencia de los mejores vistos hasta la fecha
- Facilidad de implementación
- Nombre obtenido del número de capas que lo componen



VGG16: configuración de entrenamiento

Tras las pruebas realizadas con la red anterior, se ha fijado al total el número de capas entrenadas y se buscará optimizar el modelo en función del ratio de aprendizaje.

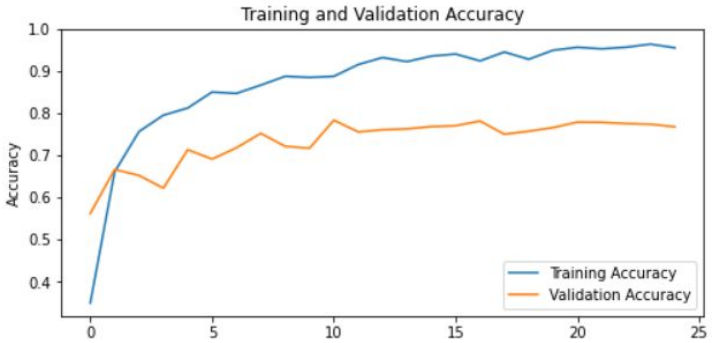
Para el resto de parámetros:

- Epochs: 25
- Optimizador adam
- Activación softMax
- Tamaño batch: 64

VGG16: resultados

Vemos que se producen unos resultados similares manteniendo el ratio de aprendizaje, y que, si bajamos el ratio demasiado, se produce pérdida de conocimiento.

Ratio de aprendizaje	Accuracy
0.0001	0.91
0.0001	0.88
0.00001	0.81



VGG16: conclusiones

Tras obtener unos resultados similares a los obtenidos con ResNet50, se buscará una tercera red con el objetivo entender las limitaciones del Transfer Learning con los parámetros que se están modificando.

Red usada	Mejor resultado
ResNet50	0.94
Inception	0.92

Xception: configuración de entrenamiento

Como última red para realizar pruebas se ha seleccionado Xception debido a los buenos resultados que suele arrojar. Para la fase de entrenamiento se ha realizado el mismo estudio que con VGG16:

- Epochs: 25
- Optimizador adam
- Activación softMax
- Tamaño batch: 64

Xception: resultados

En general, obtenemos unos resultados prácticamente idénticos a lo obtenidos con VGG16.

Learning rate	Accuracy
0.001	0.80
0.0001	0.913
0.00001	0.924

Xception: conclusiones

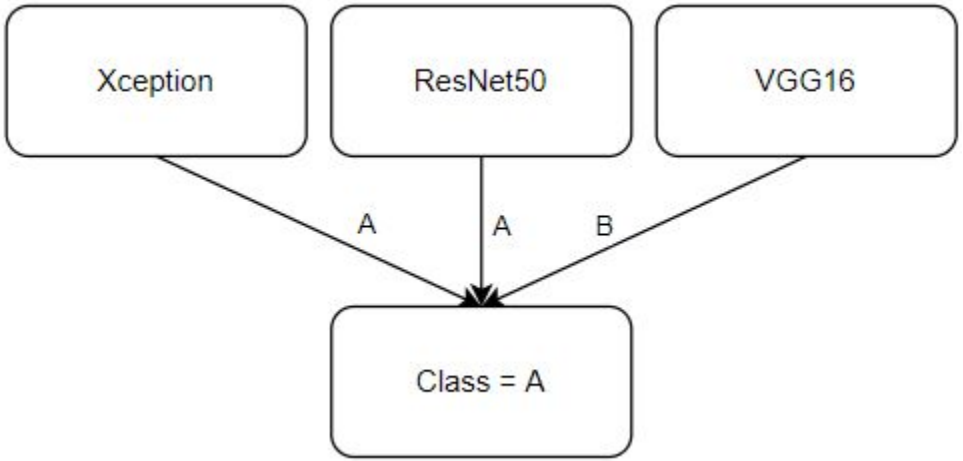
En vista de los resultados, similares a los obtenidos con los otros modelos, vemos donde reside la limitación del transfer learning rate sin aplicar ningún tipo de data augmentation ni dropout.

Mejor modelo	Resultado
Xception	0.924
VGG16	0.918
ResNet50	0.9398

Ensemble

El último paso será probar un ensemble del mejor modelo obtenido para cada red usada.

La configuración será no ponderada sobre 3 modelos con voto de clase mayoritario.



Ensemble: resultados y conclusiones

Se ha obtenido un mejor modelo con la configuración establecida. Es posible que exista mejora si se aumenta el número de modelos usados para el ensemble.

Mejor modelo	Resultado
Xception	0.924
VGG16	0.918
Resnet50	0.9398
Ensemble	**0.955**

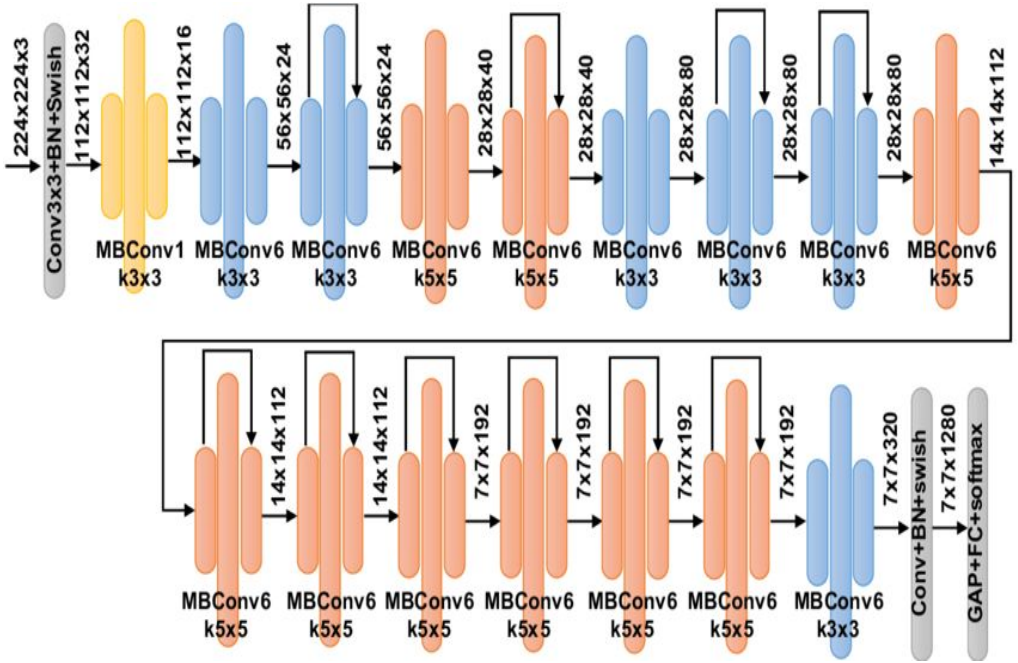
Experimentación Alejandro Pérez Lara
Usuario en kaggle: Alejandro PL

Lectura y preprocesamiento de las imágenes

- ***ImageDataGenerator***
  - Lectura de las imágenes de las carpetas según se encuentran distribuidas en las carpetas.
  - Para ciertas redes escalamos los valores para que tomen un número entre 0 y 1.
  - Reservamos entre un 20% y un 4% del conjunto de entrenamiento para realizar validación tras cada época.
- ***Data Augmentation:***
  - Trabajamos con las 4 rotaciones de las imágenes.
- ***Capa de preprocesamiento específica para cada red.***

Modelo EfficientNet V1 y V2

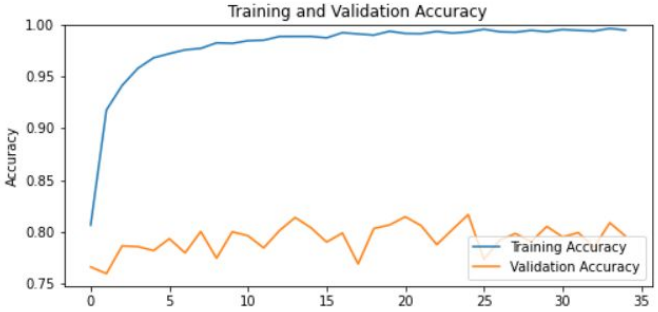
- **EfficientNetV1 (AutoKeras):**
  - Se basa en el escalado de las redes convolucionales intentando encontrar un equilibrio entre la anchura, la profundidad y la resolución
  - El número de capas oscila entre 237 y 813
- **EfficientNetV2:**
  - Optimización de la velocidad de entrenamiento y los parámetros a través de Neural Architecture Search.
  - Se introduce el entrenamiento progresivo.



EfficientNet

- **AutoKeras:** Máximo de 10 modelos con 40 épocas como máximo.
- **EfficientNetV2L:**
  - L1L2 Kernel Regularizer, L2 Bias Regularizer, L2 Activity Regularizer.
  - Early Stopping.
  - Optimizador Adam con función de pérdida categorical crossentropy .
  - FineTunning a partir de la capa 250.

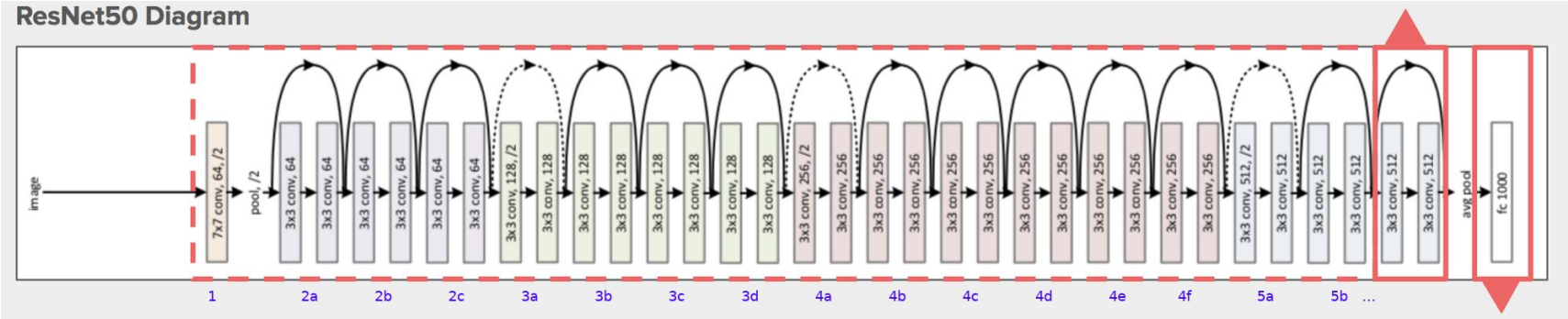
Modelo	Resultado
AutoKeras	0.78762
EfficientNetV2S	0.91237



Nota: No se pudieron probar más configuraciones para EfficientNetV2 porque hasta el última día no conseguí que funcionara debido a que la red no acepta imágenes con escala entre 0 y 1.

Modelo: Big Transfer (BiT)

- **Arquitectura base ResNet50**
  - Entrenada sobre ImageNet-21k (un conjunto mayor que ImageNet).
  - El rendimiento del modelo aumenta conforme aumenta el tamaño de los datos.
  - Mayor número de épocas que un entrenamiento estándar sobre ImageNet (90 épocas).



BiT: configuraciónes de entrenamiento

- Uso de entre 15 y 50 épocas.
- Tamaño del batch de 16.
- Uso del optimizador Adam con learning rate entre 0.001 y 0.00008.
- Función de activación softmax.
- Debido a que no se pudo hacer finetuning en algunos casos se han añadido más capas totalmente conectadas.
- Uso de data augmentation rotando las imágenes.
- Reentrenamiento de la misma red sobre distintos conjuntos de datos. Por ejemplo sobre los datos originales y un conjunto rotado del mismo tamaño.

Resultados para BiT:

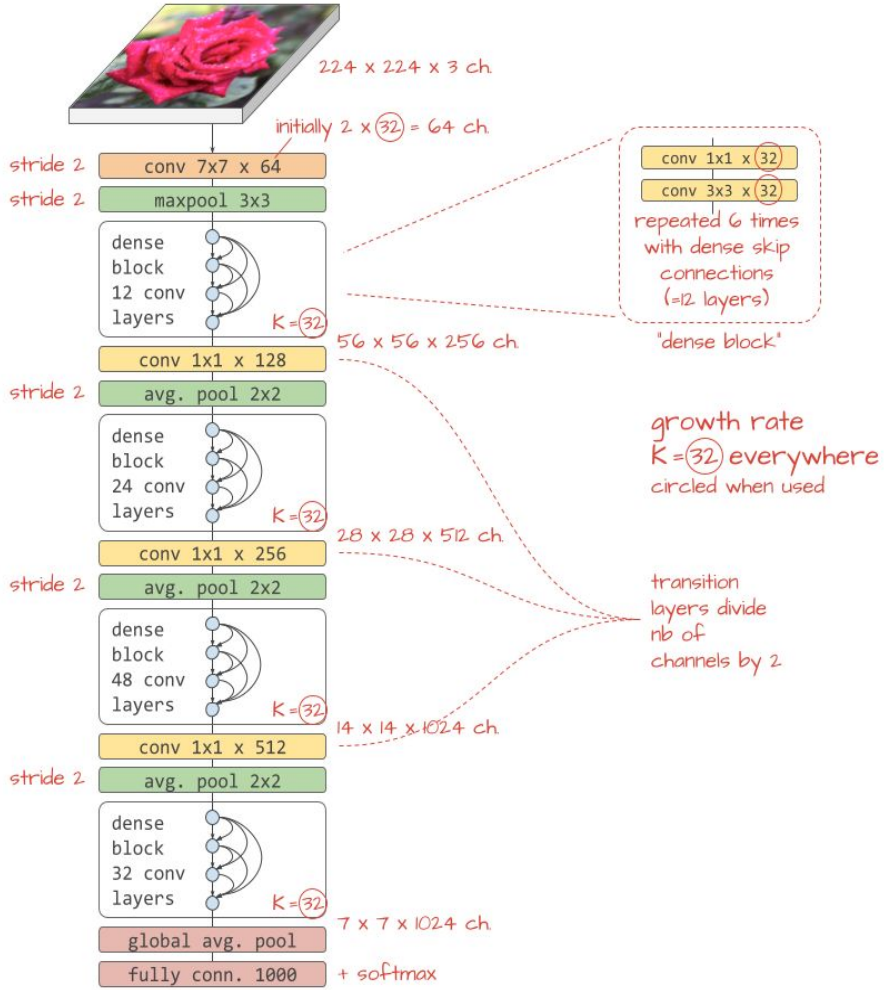
Características	Épocas	Learning rate	Private Score	Public Score
20% validación	10	0.001	0.8	0.798
20% validación	40	0.001	0.833	0.836
Sin validación	30	0.001	0.843	0.874
Datos originales + datos con una rotación de 90º (20% validación)	45	0.001	0.87	0.874
Reentrenamiento sobre datos originales + datos con alguna rotación (20% validación)	50	0.001	0.875	0.89
Igual que el anterior	65	0.001	0.875	0.89
Datos originales + datos con una rotación de 90º	15	0.00008	0.879	0.892

Conclusiones para BiT

- Mejoramos el modelo al:
  - Introducir más datos.
  - Reducir el learning rate.
- No se aprecia mejora al:
  - Introducir más capas totalmente conectadas.
  - Aumentar el número de épocas por encima de 30.

Modelo DenseNet121

- La entrada del siguiente dense block es la concatenación de todos los los dense blocks anteriores.
- Se compone de 4 dense blocks.
- Posee un total de 427 capas.
- Con esta arquitectura podemos escalar en profundidad sin preocuparnos que al hacer backpropagation los pesos apenas varíen.



DenseNet121: configuraciónes de entrenamiento

- Uso de entre 3 y 25 épocas y de early stopping para evitar el overfitting.
- Uso del optimizador Adam con un learning rate de 0.0003.
- Fine tuning a partir de la capa 100 (pesos iniciales imagenet).
- Función de activación softmax, regularización de kernel con L1, L2 y L1L2, bias regularizer con L2 y activity regularizer con L2.
- Entrenamiento sobre el conjunto de datos original y el ampliado, formado por 40000 imágenes (todas las posibles rotaciones del original).
- Uso de la capa de preprocesamiento de DenseNet.
- Añadir una capa de pooling previa a la capa totalmente conectada.

DenseNet121	Epochs	Private Score	Public Score
Modelo base (20% val.)	25	0.813	0.808
Fine tuning (20% val.)	25	0.92	0.922
Fine tuning (20% val.)	25	0.901	0.916
Fine tuning + data aug.	29	0.915	0.926
Fine tuning + data aug. + preprocess layer (1) (20% val.)	13	0.899	0.929
Fine tuning + data aug. + preprocess layer (20% val.)	7	0.917	0.943
Fine tuning + data aug. + preprocess layer (20% val.)	8	0.937	0.937
Fine tuning + data aug. + preprocess layer	**3**	**0.958**	**0.945**
(1)+ regularización L1L2 (20% val.)	7	0.944	0.942
(1) + early stopping + pooling average (20% val.)	-	0.946	0.951
(1) + early stopping + pooling average (4% val.)	-	0.954	0.956

Mejor resultado:

Obtenemos el mejor resultado al hacer preprocesamiento de las imágenes, junto con fine tuning a partir de la capa 100 y al utilizar cerca de 40000 imágenes que se corresponden al conjunto de datos original y sus 4 rotaciones posibles con una puntuación de **0.958**, el cuál se corresponde con la **séptima posición** según el ranking actual.

Por lo que hemos visto las mejoras más sustanciales se obtienen al entrenar sobre un gran conjunto de datos durante la menor cantidad de épocas posibles para evitar el sobreajuste.

Mejor resultado según Kaggle:

El mejor resultado para Kaggle se obtiene al entrenar sobre el conjunto aumentado de imágenes, usando la capa de preprocesamiento de DenseNet y con una capa de average pooling a la salida del modelo, junto con early stopping.

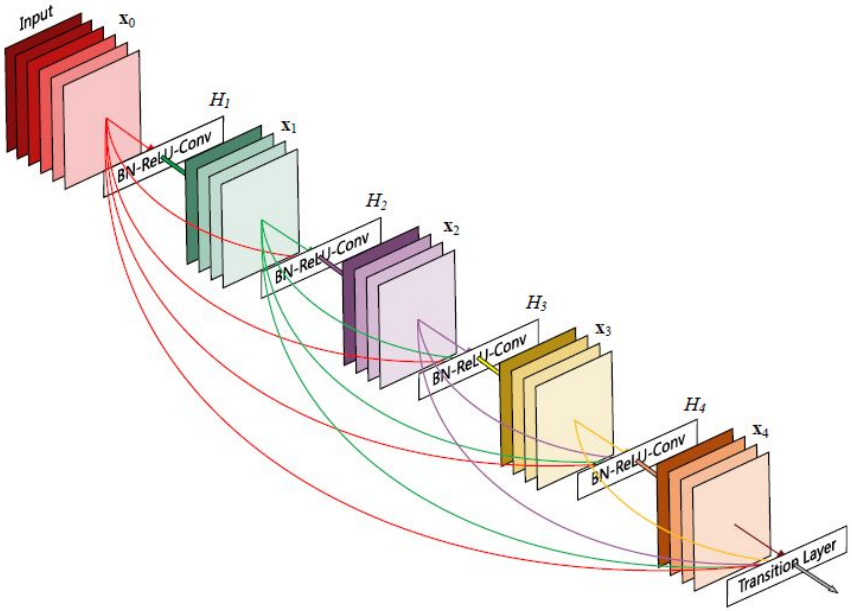
Experimentación Marcos Hernández Rodríguez
Usuario en kaggle: Marcos Hernández

Lectura y preprocesamiento de las imágenes

- ***ImageDataGenerator***
  - Lee las imágenes de la jerarquía de carpetas original para transformarlas a matrices que se pueden utilizar directamente.
  - Escala los valores de las matrices en un rango de valores entre 0 y 1.
  - Reserva un 10% del conjunto de entrenamiento para realizar validación tras cada época.
- ***Maximización de resultados con el conjunto base de imágenes***
- ***Búsqueda de hiperparametros mediante fine-tuning***

Modelo DenseNet201

- Propuesto originalmente en 2018
- Compuesto por 5 capas
- Tasa de crecimiento de k = 4.
- Cada capa toma como entrada todos los mapas de características anteriores.



DenseNet201: configuraciones de entrenamiento

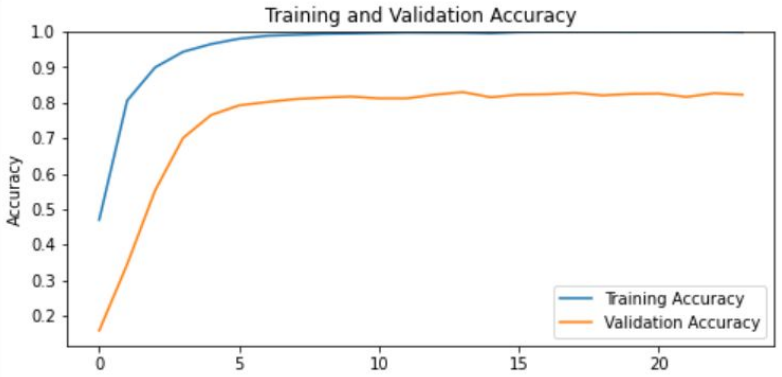
- Uso de diferentes configuraciones en el número de capas entrenadas (ninguna, toda la red, a partir de un número determinado)
- Se cargan los pesos de ImageNet
- Estudio del efecto de las épocas usadas entre 5 y 100
- Modificación del ratio de aprendizaje entre 1e-3 y 1e-5
- Optimizador: Adam
- Función de activación softMax
- Tamaño de batch entre 16 y 128

Modelo DenseNet201: Técnicas de regularización

- ***Early Stopping***
  - Se espera 10 épocas a que la métrica mejore
  - Se monitoriza el accuracy sobre el conjunto de validación
  - Se restauran los pesos del modelo a partir de la época con el mejor valor de accuracy
- ***L1 Regularization***
- ***L2 Regularization***
- ***L1 and L2 Regularization***

Modelo DenseNet201: Resultados

Model	DenseNet201
Batch size	64
Learning rate	1e-5
Re-entrenamiento de capas	All
Epochs	100
Regularization Technique	EarlyStopping Monitor: val_accuracy Patience: 10
Accuracy	**0.94432**



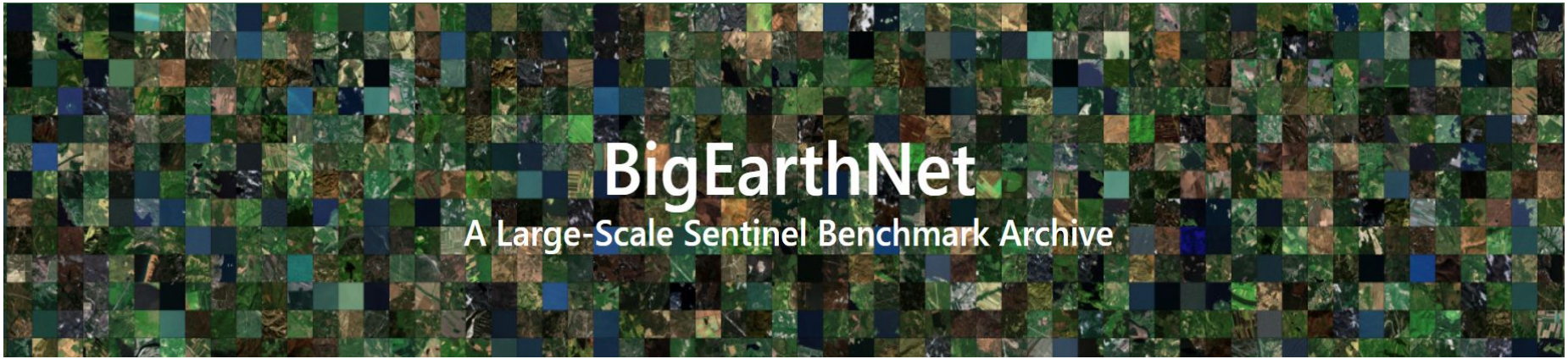
Experimentación David Murcia Gómez
Usuario en kaggle: David Murcia Gómez

Lectura y preprocesamiento

- Las imágenes se han leído con tf.keras.utils.image_dataset_from_directory() de forma que no las carga todas en memoria directamente y son más fáciles de manejar.
- En cada caso, las imágenes han recibido el escalado adecuado al input esperado por cada red neuronal.
- En todos los casos se ha tomado un 20% de las imágenes para validación, dejando un 80% para entrenamiento.

Resnet50: Big Earth Net

- Arquitectura Resnet clásica (anteriormente explicada)
- Pre entrenada con la base de datos Big Earth Net. Esta base de datos consiste en un conjunto de unas 600 mil imágenes de 10 países diferentes tomadas por satélite, similares a las de nuestro problema.



Resnet50 Big Earth Net : Configuraciones en entrenamiento

- En todos los casos se ha entrenado la última capa fully conected
- Se han probado técnicas como validación cruzada o data augmentation
- Nº variable de épocas entre 35 y 60. Con más épocas tiende al sobreajuste.
- Learning rate entre 1e-4 y 1e-5
- BatchSize = 32
- Optimizador Adam

Resultados Resnet50: Big Earth Net

Modelo	Resnet50	Resnet50	Resnet50	Resnet50
Nº Epochs	50	50	50	35
Cross Validation	Sí	Sí	No	No
Learning Rate	1e-5	1e-4	1e-4	1e-4
Data Augmentation	Sí	No	Sí	No
F1 score	0.79	0.82	0.80	0.78

Red tomada de un artículo en Towars Data Science

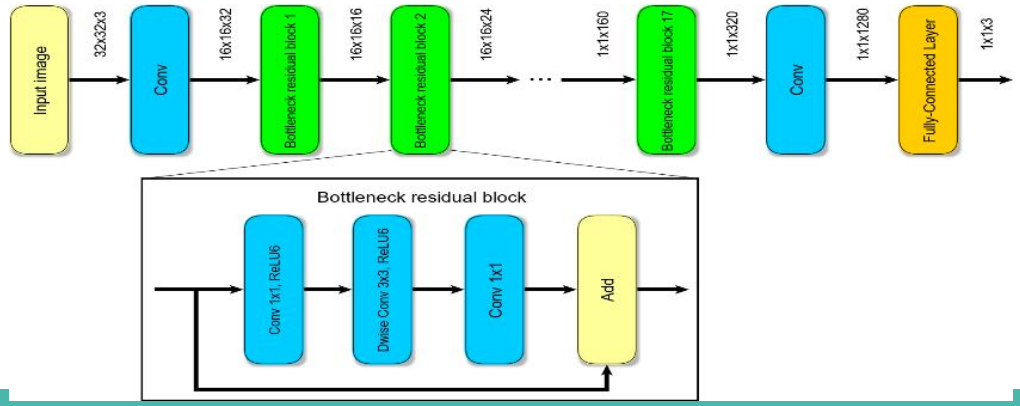
- Empleada para un problema de clasificación similar con muy buenos resultados.
- Sin embargo para nuestro problema funciona bastante mal.
- Con las mismas condiciones del modelo Resnet, da un F1 Score de 0.67.

Layer (type)	Output Shape	Param #
conv2d_18 (Conv2D)	(None, 224, 224, 28)	784
activation_24 (Activation)	(None, 224, 224, 28)	0
conv2d_19 (Conv2D)	(None, 224, 224, 28)	7084
activation_25 (Activation)	(None, 224, 224, 28)	0
max_pooling2d_9 (MaxPooling 2D)	(None, 112, 112, 28)	0
conv2d_20 (Conv2D)	(None, 112, 112, 56)	14168
activation_26 (Activation)	(None, 112, 112, 56)	0
conv2d_21 (Conv2D)	(None, 112, 112, 56)	28280
activation_27 (Activation)	(None, 112, 112, 56)	0
max_pooling2d_10 (MaxPoolin g2D)	(None, 56, 56, 56)	0
conv2d_22 (Conv2D)	(None, 56, 56, 112)	56560
activation_28 (Activation)	(None, 56, 56, 112)	0
conv2d_23 (Conv2D)	(None, 56, 56, 112)	113008
activation_29 (Activation)	(None, 56, 56, 112)	0
max_pooling2d_11 (MaxPoolin g2D)	(None, 28, 28, 112)	0
flatten_3 (Flatten)	(None, 87808)	0
dense_6 (Dense)	(None, 784)	68842256
activation_30 (Activation)	(None, 784)	0
dropout_3 (Dropout)	(None, 784)	0
dense_7 (Dense)	(None, 29)	22765
activation_31 (Activation)	(None, 29)	0

Total params: 69,084,905
Trainable params: 69,084,905
Non-trainable params: 0

MobileNetV2

- Con 53 capas es una red con muchos menos parámetros (uns 4 millones frente a los 25 millones de la Resnet50 o los 70 millones de la red del artículo de Towards Data Science)
- Preentrenada con Imagenet.
- Entrenamos sólo la última capa.
- La gran ventaja de esta red es su rapidez en entrenamiento debido al menor número de parámetros.



Epochs	20
learning rate	0.0001
Cross validation	No
Data Augmentation	No
F1 score	0.82

Bibliografía común

Xception: Deep Learning with Depthwise Separable Convolutions, François Chollet.

Xception: Implementing from scratch using Tensorflow, Arjun Sarkar, 2020.

Convolutional Neural Network and Regularization Techniques with TensorFlow and Keras, Ahmad Omar Ahsan, 2020.

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION, Diederik P. Kingma, Jimmy Lei Ba

Aatila Mustapha et al 2021 J. Phys.: Conf. Ser.

Rethinking the Inception Architecture for Computer Vision, Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna

Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi

Bibliografía común

Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation, Long D. Nguyen, Dongyun Lin, Zhiping Lin, Jiuwen Cao

Deep Residual Learning for Image Recognition, *Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun*; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778

Very Deep Convolutional Networks for Large-Scale Image Recognition, Karen Simonyan, Andrew Zisserman

What is VGG16? — Introduction to VGG16, Tinsy John Perumanoor

Documentación de Keras para DenseNet

Densely Connected Convolutional Networks

Bibliografía común

Ejemplo de implementación de BiT

Tan, M., & Le, Q. V. (2021). EfficientNetV2: Smaller Models and Faster Training. ArXiv. https://doi.org/10.48550/ARXIV.2104.00298

Documentación de Keras para EfficientNetV2

Documentación de Keras sobre los regularizadores

https://tfhub.dev/