



**Tecnológico  
de Monterrey**

## **Visión para Robots**

**Proyecto Final:**

**Inteligencia Artificial**

**Visión por Computadora**

**Miguel Marines**

## 1. Introducción

El objetivo de este proyecto es hacer una investigación sobre machine learning, especialmente deep learning, e implementar una red neuronal para poder procesar imágenes. Este proyecto procesará imágenes mediante el uso de la arquitectura ResNet34 para una red neuronal convolucional, la cual será modificada mediante la técnica de fine tuning de transfer learning. El proyecto está enfocado en detectar 6 situaciones de emergencia (riesgo de inundación, incendio, accidente automovilístico, robo de auto, altercado) y producir su clasificación y departamento de respuesta.

## 2. Machine Learning

La rama de la inteligencia artificial denominada como “machine learning” entrena sistemas computacionales para analizar datos, identificar patrones y aprender de ellos para tomar decisiones o hacer predicciones sobre nuevos datos.

La característica principal de “machine learning” es que en lugar de depender de instrucciones explícitas, sus algoritmos aprenden de grandes cantidades de datos para generalizar patrones y hacer inferencias inteligentes.

Esta área de la inteligencia artificial ha revolucionado la forma en que se procesa la información y se toman las decisiones, ofreciendo herramientas poderosas para extraer información valiosa de conjuntos de datos complejos. Adicionalmente, esta tecnología tiene aplicaciones en varios campos, incluyendo procesamiento de imágenes, procesamiento de lenguaje natural, desarrollo de modelos, etc.

Esta rama se divide en tres campos, los cuales corresponden a:

- Supervised Learning: Es un enfoque de machine learning donde un algoritmo se entrena utilizando un conjunto de datos etiquetados, con el objetivo de aprender a predecir las etiquetas de salida correctas para nuevos datos de entrada.
- Reinforcement Learning: Es un enfoque de machine learning donde un agente aprende a tomar decisiones óptimas en un entorno dinámico mediante la interacción continua con su entorno. El agente recibe recompensas o castigos según sus acciones para poder aprender.
- Unsupervised Learning: Es un enfoque de machine learning donde no se proporcionan datos etiquetados a un algoritmo. En cambio, el algoritmo busca descubrir patrones y estructuras ocultas en los datos sin ninguna guía explícita. Se utiliza principalmente para la agrupación de datos, la reducción de la dimensionalidad y la generación de nuevas representaciones de datos.

## 3. Deep Learning (DL)

Deep learning es una subrama de machine learning que se centra en entrenar redes neuronales artificiales con múltiples capas, conocidas como redes neuronales profundas. Las redes

neuronales están inspiradas en la estructura y funcionamiento del cerebro humano, donde la información se procesa a través de capas interconectadas de neuronas.

Estas redes profundas pueden aprender y extraer automáticamente características complejas de los datos, lo que las hace adecuadas para tareas que involucran grandes conjuntos de datos, como imágenes, voz y texto. El deep learning ha logrado avances significativos en áreas como el procesamiento de imágenes y el procesamiento de lenguaje natural, demostrando un gran potencial en la resolución de problemas complejos y la toma de decisiones basada en datos.

#### **4. Fundamentos de Visión por Computadora**

Se enfoca en desarrollar algoritmos y técnicas para que las máquinas puedan comprender y analizar imágenes y videos de manera similar a la visión humana. Los fundamentos clave de la visión por computadora incluyen:

- Preprocesamiento de Imágenes: Implica la manipulación de imágenes para mejorar la calidad, reducir el ruido, ajustar el contraste, etc. Se realiza a fin de facilitar el análisis y la extracción de características.
- Detección y Segmentación de Objetos: Consiste en identificar y localizar objetos de interés en una imagen o video, así como separarlos del fondo o de otros objetos presentes en la escena.
- Extracción de Características: Implica la identificación y descripción de características relevantes en las imágenes, como bordes, esquinas, texturas, colores, formas, etc. Ayuda a distinguir y clasificar objetos.
- Aprendizaje Automático: Es el uso de algoritmos de machine learning, como redes neuronales convolucionales (CNN) y clasificadores, para entrenar modelos capaces de reconocer y clasificar objetos en imágenes.
- Reconocimiento y Seguimiento de Objetos: Consiste en identificar y rastrear objetos en movimiento a través de secuencias de imágenes o videos.
- Interpretación y Comprensión de Escenas: Implica comprender el contenido y la estructura de una escena visual, incluyendo la identificación de objetos, la estimación de su posición en el espacio tridimensional y la comprensión del contexto.

Estos fundamentos de la visión por computadora se pueden aplicar a una amplia gama de áreas donde el análisis de imágenes y videos desempeña un papel fundamental.

#### **5. API's - Librerías - Plataformas - Frameworks**

Existen diversas opciones para la implementación de machine learning, entre ellas están:

- TensorFlow: Es una biblioteca de código abierto desarrollada por Google. Proporciona una plataforma completa para construir y entrenar modelos de aprendizaje automático, especialmente para redes neuronales. Ofrece una amplia gama de herramientas y funcionalidades, incluyendo el desarrollo de modelos en Python y su implementación en diferentes entornos.
- PyTorch: Es una biblioteca de aprendizaje automático de código abierto desarrollada principalmente por Facebook. PyTorch es conocido por su enfoque dinámico y flexible para construir modelos de aprendizaje automático. Es ampliamente utilizado en investigación y ofrece una amplia gama de capacidades para construir y entrenar redes neuronales.
- Scikit-Learn: Es una biblioteca de aprendizaje automático de código abierto para Python. Proporciona una amplia variedad de algoritmos y herramientas para realizar tareas comunes de aprendizaje automático, como clasificación, regresión, agrupación y reducción de dimensionalidad. Scikit-Learn también ofrece utilidades para la preparación y evaluación de datos.
- Keras: Es una biblioteca de aprendizaje automático de alto nivel y de código abierto que se ejecuta sobre TensorFlow. Keras permite construir rápidamente y de manera sencilla redes neuronales, lo que la convierte en una opción popular para los desarrolladores que buscan una interfaz fácil de usar para la creación y entrenamiento de modelos.
- MXNet: Es una biblioteca de aprendizaje automático escalable y de código abierto desarrollada por Apache. MXNet es conocido por su capacidad para entrenar modelos de manera eficiente en múltiples dispositivos, incluyendo CPUs, GPUs y sistemas distribuidos. También ofrece una interfaz en varios lenguajes de programación, como Python, R y Scala.

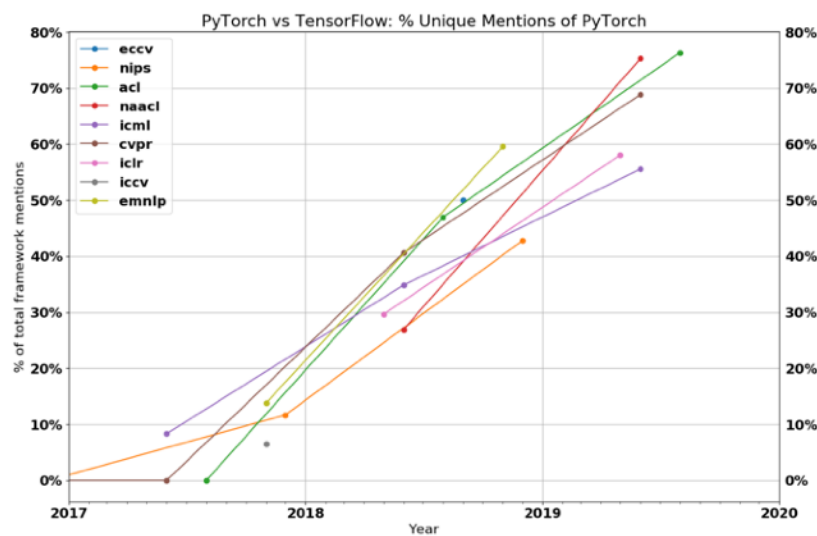
## **6. Dataset Proyecto**

El dataset contiene imágenes de una variedad de situaciones problemáticas en la calle que requieren algún tipo de asistencia. El dataset consta de 1.064 imágenes recopiladas de Internet: 250 imágenes de riesgo de inundación, 201 imágenes de incendios, 251 imágenes de protestas, 202 imágenes de accidentes automovilísticos, 47 imágenes de altercados y 113 imágenes de robos de automóviles. Todas las imágenes tienen el mismo tamaño de 300 x 300 píxeles.

## 7. Proyecto

Para la implementación de este proyecto se decidió utilizar PyTorch debido a que PyTorch ha experimentado un crecimiento significativo en su uso dentro de la comunidad de investigación de machine learning en los últimos años, lo cual implica que la mayoría de los últimos modelos e investigaciones están desarrolladas con PyTorch. Adicionalmente, PyTorch proporciona una amplia gama de herramientas y funcionalidades que permiten un desarrollo ágil y eficiente. Finalmente, PyTorch cuenta con una comunidad activa y comprometida, lo que significa que hay un amplio soporte disponible, incluyendo documentación detallada y modelos preentrenados.

La siguiente gráfica obtenida de la revista “The Gradient” muestra el uso de PyTorch en diferentes papers y conferencias de investigación.



**Figura 1:** Porcentaje de uso de PyTorch en papers y conferencias de investigación.

Link: <https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/>

La implementación del proyecto se divide en dos archivos de Python con la ayuda de PyTorch. El primer archivo (Training.py) corresponde a la implementación, ajuste, entrenamiento y guardado del modelo ResNet-34 modificado de acuerdo a las necesidades específicas del proyecto. El segundo archivo (Predictions.py) implementa el modelo ResNet-34 modificado para clasificar y hacer predicciones.

### 7.1. Arquitectura

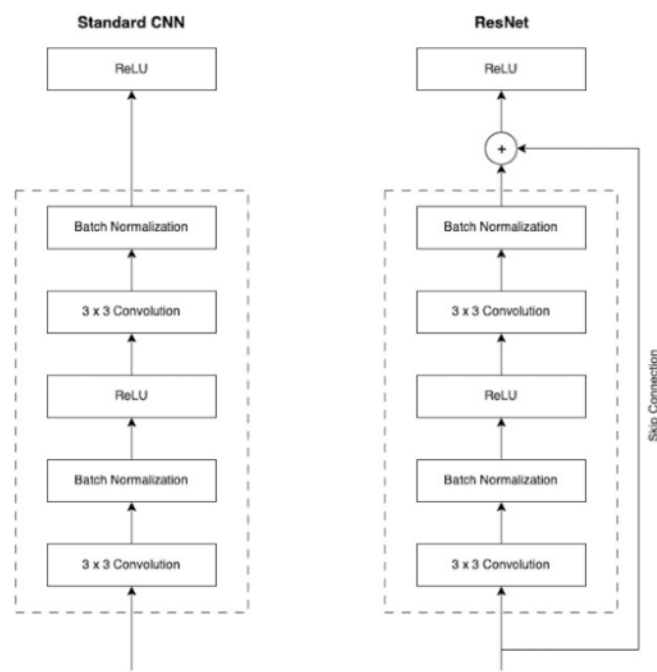
La arquitectura corresponde a la estructura del modelo de machine learning que procesa datos de entrada y produce predicciones de salida. La arquitectura incluye decisiones de diseño como el número de capas, conexiones entre capas y funciones utilizadas en la red neuronal.

## 7.2. Arquitectura de Red Neuronal Convolutacional - ResNet (Residual Neural Network)

ResNet (Residual Neural Network) es una notable arquitectura para redes neuronales convolucionales. El modelo ResNet es reconocido como uno de los mejores modelos de deep learning para el procesamiento de imágenes. En 2015 y 2016, ResNet ganó el primer lugar en ImageNet Large Scale Visual Recognition Challenge (ILSVRC), superando significativamente a otros modelos. Además, la arquitectura ResNet presenta la ventaja de eliminar el problema de vanishing gradients que tienen las redes neuronales convolucionales tradicionales.

Las redes neuronales tradicionales enfrentan el problema conocido como vanishing gradients, lo cual ocurre en el proceso de backpropagation cuando los gradientes utilizados para actualizar los parámetros de las capas anteriores en una red neuronal se vuelven extremadamente pequeños, lo que resulta en una actualización insignificante o nula de los parámetros. Como resultado, las capas anteriores de la red tienen dificultades para aprender y ajustar correctamente los parámetros. Este problema se presenta especialmente en redes neuronales profundas con muchas capas. El problema de vanishing gradients produce un rendimiento deficiente y dificulta el entrenamiento efectivo de modelos de redes neuronales profundos.

Sin embargo, para eliminar este problema, ResNet introdujo bloques residuales, que consisten en una skip connection que pasa por alto una o más capas convolucionales. Esto permite que el gradiente fluya directamente desde las capas anteriores a las capas posteriores, lo que facilita el aprendizaje y la optimización de la red. Mediante el uso de conexiones residuales, la información se conserva y los gradientes se pueden propagar de manera más efectiva, lo que permite el entrenamiento de redes neuronales profundas con mayor precisión.



**Figura 2:** Diferencia entre la arquitectura CNN estándar y la arquitectura ResNet, con la implementación de una skip connection para eliminar el problema de vanishing gradients.

En general, la arquitectura ResNet consta de múltiples capas apiladas una encima de la otra. Cada capa está compuesta por uno o más bloques residuales. Un bloque residual consta de una serie de capas convolucionales seguidas de un batch normalization y una activación de ReLU, como se muestra en la figura 2. Las arquitecturas ResNet tienen diferentes variantes, como ResNet-18, ResNet-34, ResNet-50, ResNet-101, y ResNet-152, entre otras, que se diferencian según su número de capas y parámetros.

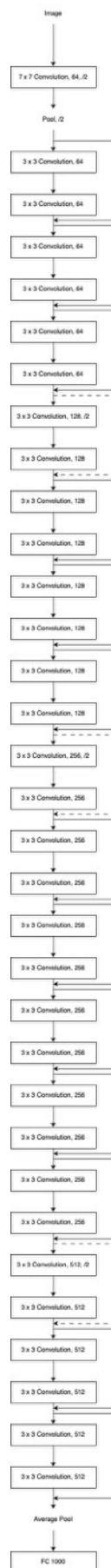
### 7.3. ResNet-34

De las diferentes variaciones de la arquitectura ResNet, se seleccionó ResNet-34 como base para el proyecto, ya que ofrece un equilibrio entre complejidad del modelo y rendimiento. Además, presenta un equilibrio entre profundidad y eficiencia computacional en comparación con arquitecturas ResNet más profundas y con más capas, ya que se tienen recursos computacionales limitados y limitaciones de tiempo para este proyecto.

Las siguientes imágenes muestran la arquitectura del modelo ResNet-34 implementado.

layer name	34-layer
conv1	
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$

**Figura 3:** Convoluciones de ResNet-34.



**Figura 4:** Arquitectura ResNet-34.



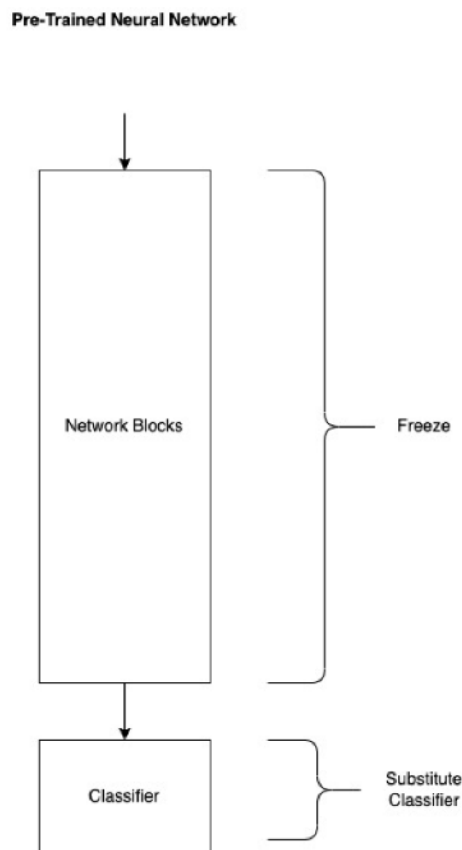
#### 7.4. Transfer Learning con Red Neuronal Convolutacional Preentrenada - ResNet

Transferir learning con ResNet es una técnica muy poderosa en machine learning que aprovecha los parámetros previamente entrenados de un modelo ResNet entrenado con el dataset de ImageNet y aplica su conocimiento aprendido a nuevas tareas o datasets con datos limitados.

La aplicación de esta técnica ayuda a reducir el tiempo de entrenamiento y los recursos computacionales requeridos. Además, ayuda a aumentar la precisión cuando se trabaja con conjuntos de datos complicados. El modelo ResNet preentrenado proporciona una base sólida para aprender características relevantes para este proyecto, lo que permite que el modelo logre una mayor precisión con menos ejemplos de entrenamiento.

#### 7.5. Transfer Learning - Fine Tuning

Fine tuning es una técnica de transfer learning en la que los primeros bloques de una red neuronal entrenada se congelan y luego se sustituye un nuevo clasificador directamente en la red neuronal. Posteriormente, solo se entrena el nuevo clasificador ya que se congelan los parámetros de los bloques anteriores.

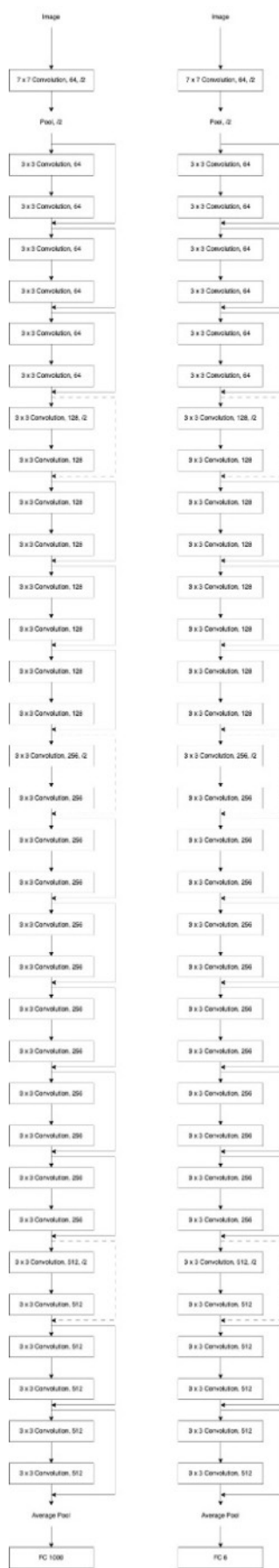


**Figure 5:** Técnica de fine tuning en la que se congelan los primeros bloques y se introduce un nuevo clasificador.

### 7.6. Aplicaron de Transfer Learning y Fine Tuning con Modelo Preentrenado ResNet-34

Para aplicar las técnicas de transfer learning previamente mencionadas, se congelaron los primeros bloques de la red neuronal entrenada ResNet-34 y sus parámetros. Posteriormente, la capa de clasificación original del modelo ResNet-34 fue reemplazada por una nueva capa igual al número de clases.

La imagen de la página siguiente muestra la modificación realizada a la arquitectura del modelo preentrenado ResNet-34 en l aplicación de la técnica de transfer learning, fine tuning.



**Figura 6:** La arquitectura de la izquierda es ResNet-34 original. La arquitectura de la derecha es ResNet-34 con la aplicación de fine tuning en el proceso de transfer learning.

## 7.7. Manejo del Accuracy y el Learning Rate

### 7.7.1. Cross-Entropy

Cross-Entropy es una función utilizada en machine learning, particularmente en clasificación. Esta función proporciona una medida cuantitativa del rendimiento de un modelo, ya que mide la disimilitud entre la distribución de la probabilidad predicha y la distribución de la probabilidad real del target. La salida de cross-entropy es un valor de probabilidad entre 0 y 1, 0 presentando ninguna pérdida y un modelo perfecto.

Esta función es de suma importancia en el proceso de entrenamiento de un modelo ya que obtiene la pérdida y ayuda a guiar el proceso de aprendizaje para minimizar la pérdida y ajustar los parámetros para realizar predicciones con mayor precisión.

### 7.7.2. Optimizador Adam

Un optimizador es un algoritmo utilizado para ajustar los parámetros de un modelo durante el proceso de entrenamiento con el objetivo de minimizar la pérdida y aumentar la precisión del modelo. Una de las principales ventajas que ofrecen los optimizadores es su capacidad para superar las limitaciones de los learning rates fijos.

Para este proyecto, se seleccionó el optimizador Adam debido a su robustez, rápida convergencia y capacidad para manejar diferentes tipos de datos de manera efectiva. El optimizador Adam es un algoritmo de optimización adaptable que combina conceptos de los algoritmos de momentum y RMSprop para lograr actualizaciones de parámetros eficientes y efectivas.

El principio fundamental del optimizador de Adam es ajustar de manera adaptativa la tasa de aprendizaje para cada parámetro en el modelo en función de mantener un promedio móvil tanto del primer momento (la media - algoritmo de momentum) como del segundo momento (la varianza no centrada - algoritmo RMSprop) de los gradientes. Al adaptar las tasas de aprendizaje en función de la magnitud y la dirección de los gradientes, el optimizador Adam puede ajustar dinámicamente los diferentes parámetros, lo que conduce a una convergencia más rápida y un mejor rendimiento de optimización.

## 7.8. Entrenamiento y Testing (Training.py)

Luego de modificar el modelo ResNet-34 para adecuarlo a las necesidades del proyecto, se dispuso de los datos de entrenamiento del dataset, que corresponde al 70% de las imágenes de cada clase, para realizar el entrenamiento de la arquitectura modificada.

El modelo ResNet-34 previamente entrenado se entrenó con el dataset ImageNet, el cual tiene imágenes del tamaño fijo de 224 x 224 píxeles. Por lo tanto, para garantizar la consistencia y facilitar el trabajo del modelo, las imágenes del dataset de este proyecto se ajustaron a un tamaño de 224 x 224 píxeles. Además, las imágenes también se normalizaron como el dataset de ImageNet para mantener la consistencia utilizando los valores de desviación estándar y media,

que corresponden respectivamente a [0.485, 0.456, 0.406] y [0.229, 0.224, 0.225] para los canales de tres colores (RGB).

Luego de asegurar la consistencia de las imágenes, el nuevo clasificador se entrenó con las nuevas imágenes proporcionadas para ajustar el modelo a las necesidades específicas del proyecto. Las imágenes para entrenar el modelo se proporcionaron en batches de 32 imágenes para cada epoch, que contienen imágenes de las diferentes clases.

Para entrenar al nuevo clasificador se utilizó un total de 15 epochs. Para guiar el entrenamiento, se utilizó la función de cross-entropy para medir la pérdida y saber qué tan bien se estaba desempeñando el modelo. El optimizador Adam se utilizó para ajustar de forma adaptativa el learning rate de cada parámetro y ayudar a encontrar los parámetros óptimos durante el proceso de entrenamiento.

En cada epoch, se realizaron 32 predicciones de acuerdo con los nuevos parámetros calculados y el batch de 32 imágenes proporcionadas. La precisión de cada epoch se midió dividiendo el número de predicciones correctas entre el número total de predicciones.

Para cada epoch específico, el programa imprimió su pérdida correspondiente y su precisión alcanzada. Además, la precisión se comparó en cada epoch con la precisión del epoch anterior para guardar los parámetros y el modelo del epoch con la mejor precisión. Además, cuando los epochs se iteraron por completo, el programa imprimió la precisión de entrenamiento del mejor epoch, el cual corresponde al modelo y los parámetros guardados.

Adicionalmente, Era importante realizar una prueba del modelo guardado para conocer la realidad de qué tan bien se estaba desempeñando el modelo y saber si se estaba haciendo overfitting. Por lo tanto, las imágenes de prueba, que corresponden al 30 % de las imágenes del dataset, se cargaron en el modelo guardado para realizar sus predicciones correspondientes. Las imágenes se redimensionaron y normalizaron como con las imágenes de entrenamiento para ser consistentes con las características de las imágenes con las que se entrenó previamente el modelo ResNet-34. La precisión de la prueba se midió dividiendo el número de predicciones correctas entre el número total de predicciones.

### 7.9. Predicciones (Predictions.py)

En el segundo archivo de Python se importó el modelo guardado previamente de ResNet-34 modificado y con sus parámetros para poder hacer predicciones sin tener que esperar al entrenamiento del modelo. Después de importar el modelo, el programa verifica un directorio específico cada 5 segundos para cambiar el tamaño y normalizar sus imágenes de acuerdo con las características del conjunto de datos de ImageNet y alimentarlas al modelo para clasificarlas y hacer las predicciones correspondientes. Las imágenes fueron proporcionadas con sus correspondientes coordenadas (nombre de la imagen) para que el sistema pueda conocer la ubicación de la situación.

## 7.10. Resultados

El modelo modificado alcanzó una precisión de 0.9906666666666667 en la fase de entrenamiento y una precisión de 0.9777070063694268 en la fase de pruebas. Dado que el modelo funciona bien tanto en el entrenamiento como en la prueba, se pueden descartar overfitting and underfitting.

La tabla de la página siguiente muestra los resultados del entrenamiento del modelo ResNet-34 modificado en el archivo “Training.py”:

**Tabla 1:** Cada fila de la tabla representa una epoch. en la primera columna se numeran los epochs; en la segunda columna se muestran los costos; y en la tercera columna se presenta el accuracy.

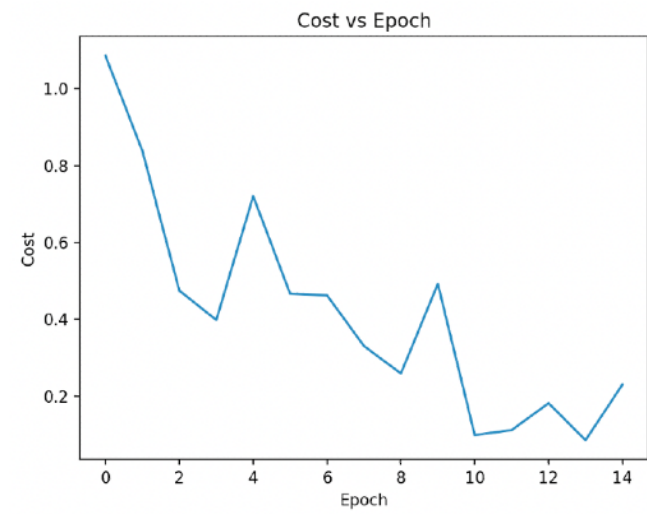
Epoch	Cost	Accuracy
0	1.085547685623169	0.7386666666666667
1	0.8376806378364563	0.8746666666666667
2	0.4745651185512543	0.9306666666666666
3	0.3982786536216736	0.9586666666666667
4	0.7202664017677307	0.9666666666666667
5	0.4664444327354431	0.9666666666666667
6	0.46225932240486145	0.976
7	0.33046668767929077	0.984
8	0.2590530514717102	0.9853333333333333
9	0.49206024408340454	0.984
10	0.09898990392684937	0.9853333333333333
11	0.11182408779859543	0.988
12	0.1817815601825714	0.9906666666666667
13	0.08545726537704468	0.9893333333333333
14	0.23032332956790924	0.9906666666666667

La siguiente tabla muestra el accuracy de entrenamiento mejor logrado y el accuracy de prueba del modelo ResNet-34 modificado en el archivo “Training.py”:

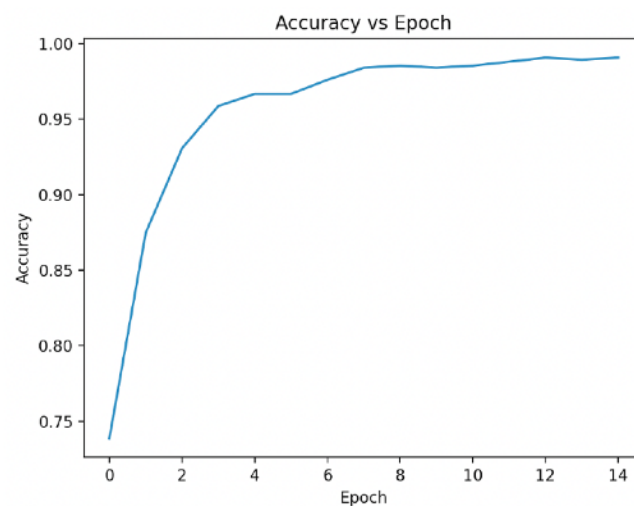
**Tabla 2:** Presenta el mejor accuracy de entrenamiento y el accuracy de prueba.

Accuracies
Best Training Accuracy: 0.9906666666666667
Testing Accuracy: 0.9777070063694268

Las siguientes imágenes muestran el comportamiento del costo y accuracy con respecto a los epochs en el entrenamiento del modelo ResNet-34 modificado en el archivo “Training.py”.









**Figura 7:** Gráfico del comportamiento del costo con respecto a los epochs.



**Figura 8:** Gráfico del comportamiento del accuracy con respecto a los epochs.

Además, se presentaron imágenes de situaciones problemáticas al archivo "Predictions.py" para clasificar las situaciones problemáticas presentadas y procesar las respuestas adecuadas.

**Tabla 3:** Presenta ejemplos de la clasificación de situaciones problemáticas. La información procesada en la predicción incluye el ID de la situación, las coordenadas de la situación presentada y las unidades de asistencia correspondientes.

Image ID	Coordinates	Units of Assistance
Image: 		
ID: 0001	Coordinates: 33.8568, 151.2153	Units of Assistance: ['Protest => Police Force and Transit Police']
Image: 		
ID: 0002	Coordinates: 55.7558, 37.6176	Units of Assistance: ['Flood Risk => Drainage Department and Transit Police']
Image: 		
ID: 0003	Coordinates: 40.6892, 74.0445	Units of Assistance: ['Fire => Firefighters and Transit Police']
Image: 		
ID: 0004	Coordinates: 37.7749, 122.4194	Units of Assistance: ['Altercation => Police Department']
Image: 		
ID: 0005	Coordinates: 27.1750, 78.0422	Units of Assistance: ['Accident => Towing Unit, Ambulance and Transit Police']
Image: 		
ID: 0006	Coordinates: 40.7128, 74.0060	Units of Assistance: ['Robbery => Police Department']



## 8. Referencias

- K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv.org, Dec. 10, 2015. [Online]. Available: <https://arxiv.org/abs/1512.03385>.
- "ImageNet Winning CNN Architectures (ILSVRC) | Data Science and Machine Learning," [www.kaggle.com](http://www.kaggle.com). [Online]. Available: <https://www.kaggle.com/getting-started/149448>.
- "Papers with Code - ImageNet Benchmark (Image Classification)," [paperswithcode.com](http://paperswithcode.com). [Online]. Available: <https://paperswithcode.com/sota/image-classification-on-imagenet>.
- R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training Recurrent Neural Networks," arXiv:1211.5063 [cs], Feb. 2013. [Online]. Available: <https://arxiv.org/abs/1211.5063>.
- Chi-Feng Wang, "The Vanishing Gradient Problem," Medium, Jan. 08, 2019. [Online]. Available: <https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484>.
- S. J. Pan and Q. Yang, "A Survey on Transfer Learning," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, Oct. 2010. doi: 10.1109/tkde.2009.191.
- Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, "SpotTune: Transfer Learning Through Adaptive Fine-Tuning," [openaccess.thecvf.com](http://openaccess.thecvf.com), 2019. [Online]. Available: [https://openaccess.thecvf.com/content\\_CVPR\\_2019/html/Guo\\_SpotTune\\_Transfer\\_Learning\\_Through\\_Adaptive\\_Fine-Tuning\\_CVPR\\_2019\\_paper.html](https://openaccess.thecvf.com/content_CVPR_2019/html/Guo_SpotTune_Transfer_Learning_Through_Adaptive_Fine-Tuning_CVPR_2019_paper.html)
- K. E. Koech, "Cross-Entropy Loss Function," Medium, Feb. 25, 2021. [Online]. Available: <https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e>.
- D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv.org, Dec. 22, 2014. [Online]. Available: <https://arxiv.org/abs/1412.6980>.