

# Apuntes de clase

Sebastián Campos Zuñiga

15 de Febrero de 2024

## 1 Tarea Aportes a la Inteligencia Artificial

Yann LeCun

- Considerado el padre del Deep learning, uso de redes neuronales, usado para el reconocimiento de voz, computer visión y procesamiento de lenguaje natural.

Yoshua Bengio

- Contribuciones claves en modelos probabilísticos de secuencias, se utilizan para reconocimiento de voz, de escritura y en modelos no supervisados.

San Altman

- Es uno de los fundadores de chat GPT, una de las mas recientes herramientas de inteligencia artificial en disposición a todo público.

Geoffrey Hinton

- Sus contribuciones van en los ámbitos de reconocimiento del habla, imágenes y procesamiento de le lenguaje natural, sus trabajos permitieron la ejecución del machine learning para desarrollar los asistentes personales y los carros sin conductor.

Timnit Gebru

- Ha investigado, sobre la discriminación en los sistemas de IA, además, de hacer conciencia sobre la ética y responsabilidad sobre el uso y desarrollo de las tecnologías de inteligencia artificial.

Ian Goodfellow

- Desarrollo técnicas que permiten a la inteligencia artificial, inventar voces, rostros y edificios. Desarrollo GAN, son dos redes neuronales que compiten entre ellas para mejore cada vez más.

## 2 Noticias

### Lumiere: A Space-Time Diffusion Model for Video Generation

El paper nos habla de los retos para la creación de videos por medio de una serie de palabras, donde se interpretan para que la inteligencia artificial genere un video acorde a lo que se escribió. Ellos indican que el enfoque nuevo que le están dando con este modelo es generar el marco de trabajo total de la duración del video de una sola vez, usando una arquitectura llamada "Space-Time U-Net", permitiendo generar hasta 80 marcos de 16 fps. La ventaja de este modelo es que les permite tener una mayor consistencia de los movimientos.

El modelo está creado utilizando tecnologías existentes, como el Text to Image, en el cual ellos construyeron su modelo. Se utiliza el modelo generativo llamado "Diffusion Probabilistic Model". El modelo puede aceptar tanto textos como imágenes para generar un video. El modelo fue entrenado con un dataset de 30 millones de videos, todos con su descripción de texto, cada video es de 5 segundos.

Ver Figura 1.

### Sora AI: Creación de videos a partir de textos, antecesores DALI AI

Sora AI es la propuesta de OpenAI para la generación de videos, por medio de texto o una imagen. El modelo utiliza los videos y las imágenes como colecciones muy pequeñas llamadas "patches", los cuales son tokens de GPT, para poder generar videos de diferentes resoluciones, duraciones y de diferentes relaciones de aspecto. Ver Figura 2 para una representación de los "patches".

Es un modelo de difusión; se le da una entrada, el está entrenado "sucio" para predecir el patch pero en "limpio". Este modelo es escalable; entre mayor capacidad computacional se le dé, mejores resultados obtendrá. Ver Figura 3.

## 3 Programación Vectorial

### Pandas

¿Qué es Pandas? Es una librería de Python especializada en trabajar con datasets. Algunas de sus funciones son analizar, limpiar, explorar y manipular datos. Permite analizar grandes cantidades de datos y hacerlos legibles y relevantes.

### Matplotlib

¿Qué es Matplotlib? Es una librería de Python para graficar datos, para su representación visual.

## NumPy

¿Qué es NumPy? Es una librería de Python para trabajar con arrays; tiene funciones especializadas para trabajar con álgebra lineal y matrices. Es hasta 50 veces más rápido que trabajar con las listas tradicionales de Python. Está en parte escrito en C y C++, además de estar optimizado para un espacio continuo en memoria.

### Creación de arrays

Creación simple de un array usando la librería NumPy:

```
array_1d = np.arange(10)
print("Array 1D:", array_1d)
```

Resultado:

```
Array 1D: [0 1 2 3 4 5 6 7 8 9]
```

Creación de array solo con 1 y 0:

```
array_1d = np.ones((10))
print("Array 1D:", array_1d)
array_1d = np.zeros((10))
print("Array 1D:", array_1d)

Array 1D: [1. 1. 1. 1. 1. 1. 1. 1. 1.]
Array 1D: [0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

### Creación de matrices

Creación de matrices utilizando NumPy. Recibe como parámetro opcional la dimensión, filas y columnas:

```
array_1d = np.identity(3)
print(array_1d)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

```
array_1d = np.ones((4, 5))
print(array_1d)
```

```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

```
array_1d = np.ones((2,4, 5))
print(array_1d)
```

```
[[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]^4
 [1. 1. 1. 1. 1.]]]
```

```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

## Operaciones Básicas

La librería permite operaciones básicas con los operadores simples: suma (+), resta (-), multiplicación (\*) y división (/).

```
# Crear dos arrays NumPy de 1D con números aleatorios
lst = [2, 4, 6]
lst2 = [1, 3, 5]
# converting list to array
array_random_1 = np.array(lst)
array_random_2 = np.array(lst2)

# Realizar operaciones element-wise
suma_resultado = array_random_1 + array_random_2
resta_resultado = array_random_1 - array_random_2
multiplicacion_resultado = array_random_1 * array_random_2
division_resultado = array_random_1 / array_random_2

# Imprimir los resultados de cada operación

print("Suma:", suma_resultado)
print("Resta:", resta_resultado)
print("Multiplicación:", multiplicacion_resultado)
print("División:", division_resultado)
print("Elevar:", array_random_1**2)
```

```
Suma: [ 3  7 11]
Resta: [1 1 1]
Multiplicación: [ 2 12 30]
División: [2.          1.33333333 1.2         ]
Elevar: [ 4 16 36]
```

## Operaciones por dimensiones

Operaciones por dimensiones. Algunos ejemplos importantes son la suma de todos los elementos (se usa `sum`), y la suma de cada columna de la matriz o cada fila:

```

▼ matriz_2d = np.array([[1, 2, 3],
| | | | | | | |
| [4, 5, 6],
| [7, 8, 9]])

print("Resultado de sumar todos los elementos de la matriz:", np.sum(matriz_2d))
print("Resultado de sumar los elementos de cada columna matriz: ", np.sum(matriz_2d, axis=0))
# matriz_2d[:, 0] , matriz_2d[:, 1] , matriz_2d[:, 2]
print("Resultado de sumar los elementos de cada fila matriz: ", np.sum(matriz_2d, axis=1))
# matriz_2d[0, :] , matriz_2d[1, :] , matriz_2d[2 :]

Python
Resultado de sumar todos los elementos de la matriz: 45
Resultado de sumar los elementos de cada columna matriz: : [12 15 18]
Resultado de sumar los elementos de cada fila matriz: : [ 6 15 24]

```

## Funciones estadísticas

Operaciones estadísticas que permiten ver el resultado de la media (usando `mean`), desviación estándar (`std`), valor máximo (`max`) y el valor mínimo (`min`):

```

array_random = np.random.rand(100)
# Calcular la media, desviación estándar, valor máximo y mínimo
media = np.mean(array_random)
desviacion_estandar = np.std(array_random)
valor_maximo = np.max(array_random)
valor_minimo = np.min(array_random)

# Imprimir los resultados de cada operación
print("Media:", media)
print("Desviación estándar:", desviacion_estandar)
print("Valor máximo:", valor_maximo)
print("Valor mínimo:", valor_minimo)

Media: 0.49849874449499715
Desviación estándar: 0.2847645198211511
Valor máximo: 0.9756109910102987
Valor mínimo: 0.009293305538057517

```

## K-Nearest Neighbor

El algoritmo K-Nearest Neighbor se utiliza para clasificación. Dada una nueva instancia, con una distancia  $k$ , se revisan los vecinos más cercanos que comparten las mismas características para asignar su clase. Esto puede determinarse por moda o por cercanía utilizando la distancia euclíadiana.

Función para calcular la distancia euclíadiana con NumPy, mucho más eficiente que con ciclos:

```

# Función de programación vectorial con NumPy para calcular la distancia euclídea
def distancia_euclidiana_vectorial_explicita(vector1, vector2, printTime = True):
    inicio = time.time()

    # Elevar al cuadrado cada elemento de la diferencia
    diferencia_cuadrada = (vector1 - vector2)**2

    # Sumar los elementos cuadrados
    suma_cuadrada = np.sum(diferencia_cuadrada)

    # Calcular la raíz cuadrada de la suma
    resultado = np.sqrt(suma_cuadrada)

    fin = time.time()
    tiempo_total = fin - inicio
    if printTime:
        print(f"Tiempo vectorial explícito: {tiempo_total:.6f} segundos")
    return resultado

```

## References

- [1] Omer Bar-Tal, Hila Chefer, et al. *Lumiere: A Space-Time Diffusion Model for Video Generation*. Google Research, 2024.
- [2] OpenAI. (s.f.). Video Generation Models as World Simulators. Recuperado de <https://openai.com/research/video-generation-models-as-world-simulators>
- [3] w3schools. (s.f.). Pandas Introduction - w3schools. Recuperado de [https://www.w3schools.com/python/pandas/pandas\\_intro.asp](https://www.w3schools.com/python/pandas/pandas_intro.asp)
- [4] w3schools. (s.f.). Matplotlib Pyplot - w3schools. Recuperado de [https://www.w3schools.com/python/matplotlib\\_pyplot.asp](https://www.w3schools.com/python/matplotlib_pyplot.asp)

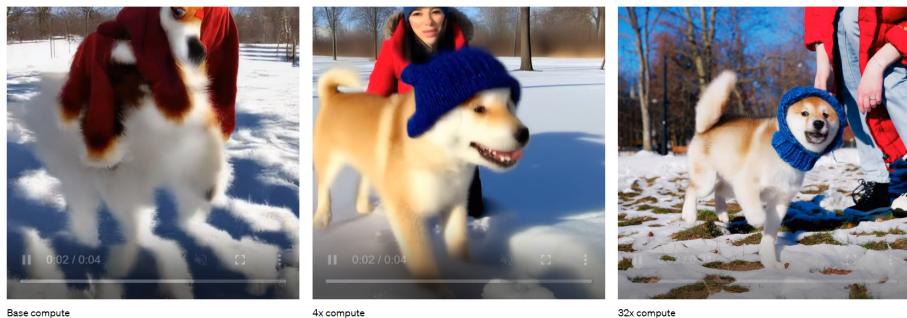


Figure 3: Ejemplo de imagen con diferente poder computacional

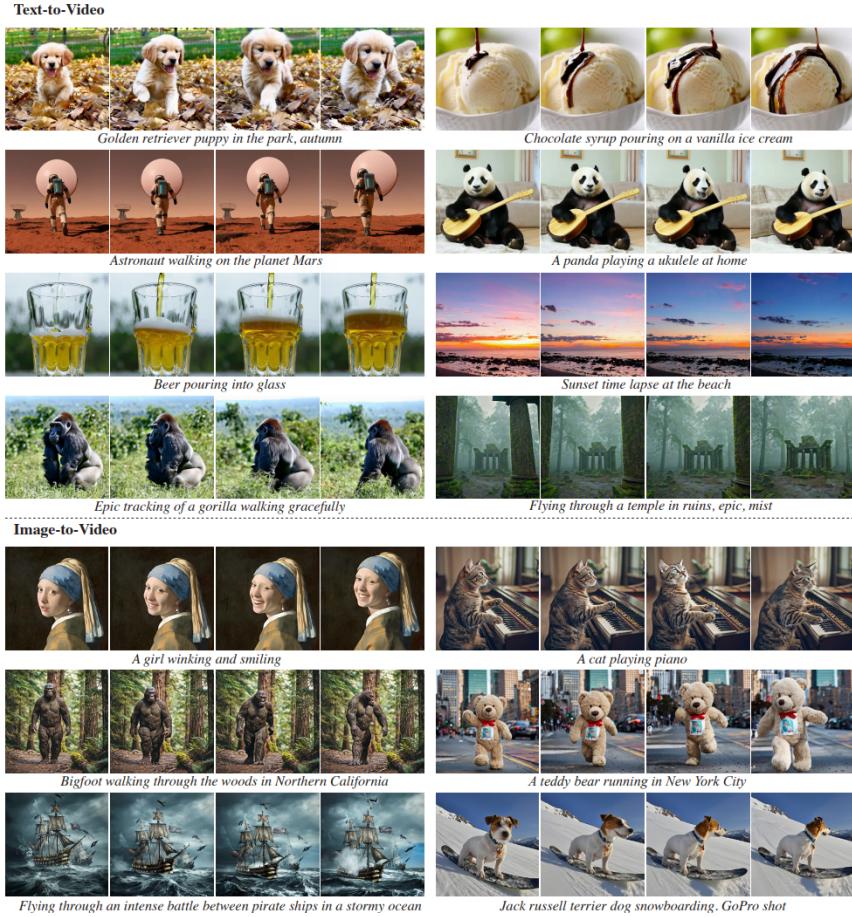


Figure 1: Ejemplo de generación de video basado en texto y una imagen en específico.

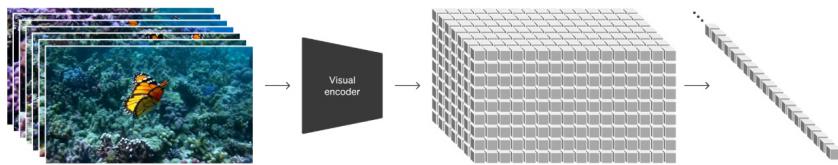


Figure 2: Ejemplo patches en Sora IA.