

# Curso Redes Neuronales

Profundización en modelos y técnicas avanzadas

Sergio Barrachina   Nacho Mestre

January 16, 2025

1. Repaso
2. Overfitting
3. Transformer
4. Modelos prediseñados y preentrenados
5. Práctica

- Inicialización (Automático)
- Entrenamiento
  1. Cargar muestras (por lotes)
  2. Pasar las muestras por la red y obtener una predicción → Forward-Pass
  3. Comparar con el resultado esperado → Función de pérdida
  4. Calcular el gradiente de cada parámetro → Backward-Pass (AutoGrad)
  5. Actualizar los parámetros → Optimizador
  6. Volver al punto 1
- Inferencia

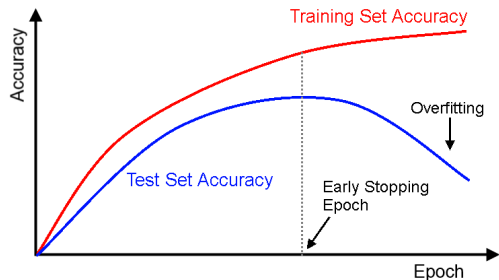
## **Con parámetros entrenables:**

- Densa, Fully-Connected o Linear
- Convolutacional
- Batch Normalization

## **Transformaciones y activaciones:**

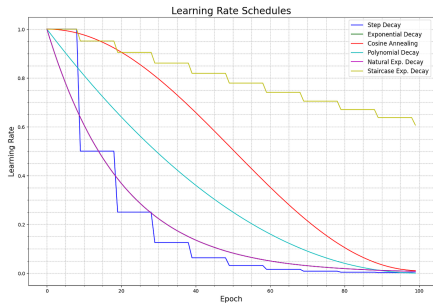
- Pooling (junto a convoluciones)
- ReLU (activación)
- Softmax
- Flatten (aplanado)

# Overfitting



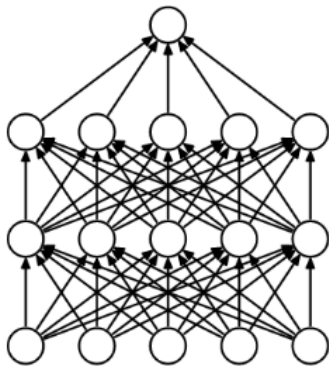
```
1 if val_loss < best_val_loss:
2     best_val_loss = val_loss
3     best_model_weights =
4         model.state_dict()
5     epochs_no_improve = 0
6 else:
7     epochs_no_improve += 1
8 if epochs_no_improve >= patience:
9     break
```

# Learning Rate variable

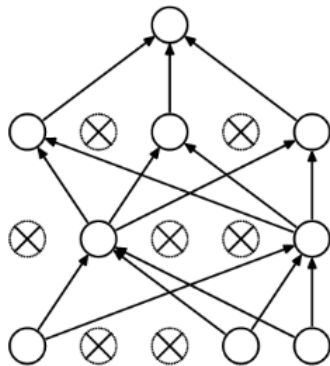


```
1 for epoch in range(n_epochs):
2     cosine_decay = 0.5 * (1 + cos(pi *
3         epoch / num_epochs))
4     new_lr = lr_min + (initial_lr -
5         lr_min) * cosine_decay
6     for param_group in
7         optimizer.param_groups:
8         param_group['lr'] = new_lr
```

# Capas Dropout

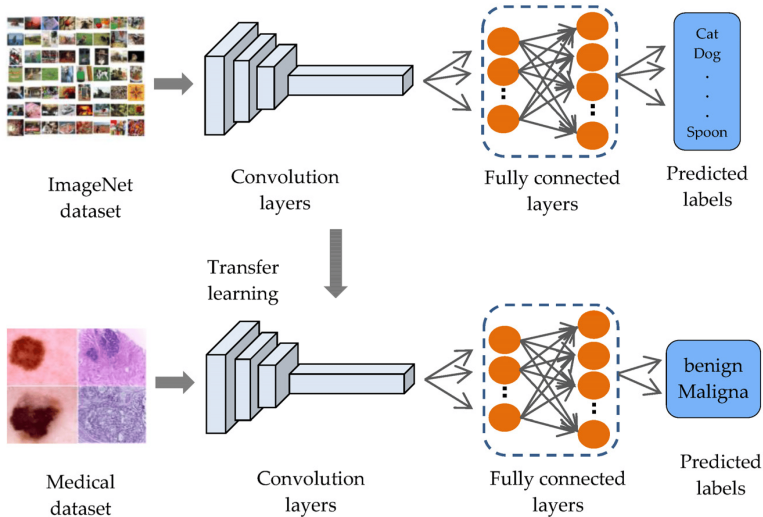


(a) Standard Neural Net



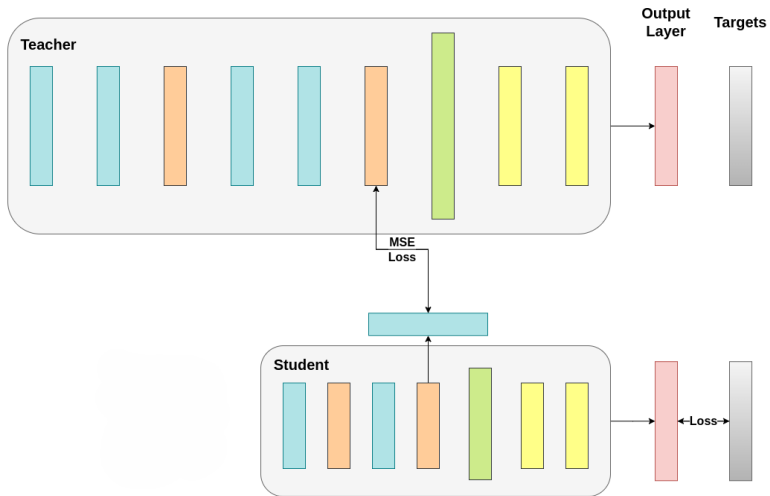
(b) After applying dropout.

# Transferencia de conocimiento





# Destilar el conocimiento

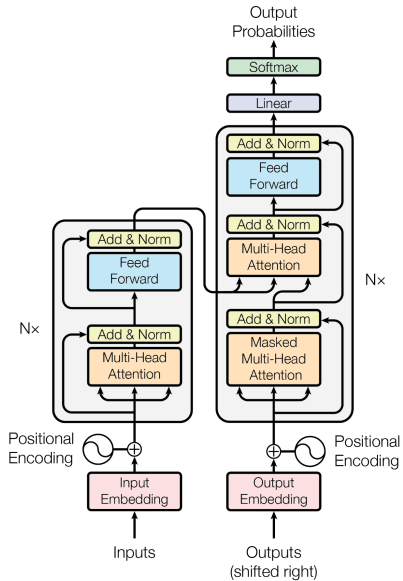


---

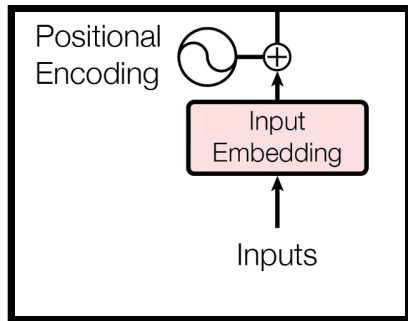
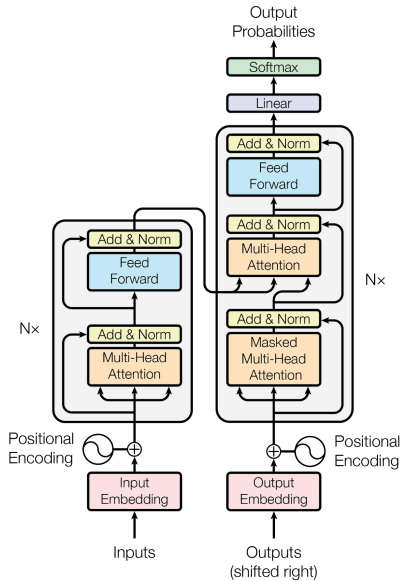
## **Attention Is All You Need**

---

# Transformer



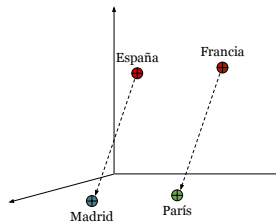
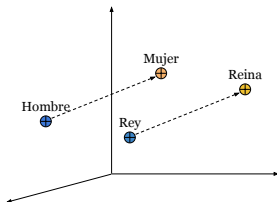
# Transformer



# Embeddings

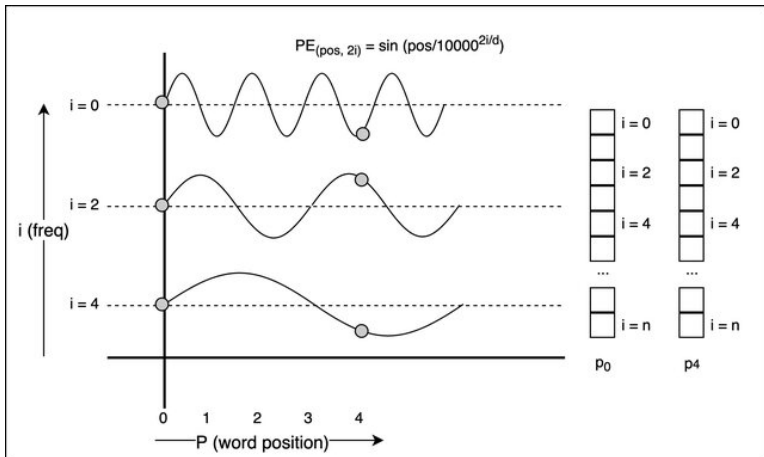
<i>man</i>	→	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
<i>woman</i>	→	0.7	0.3	0.9	-0.7	0.1	-0.5	-0.4
<i>king</i>	→	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6
<i>queen</i>	→	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9

Word                      Word embedding



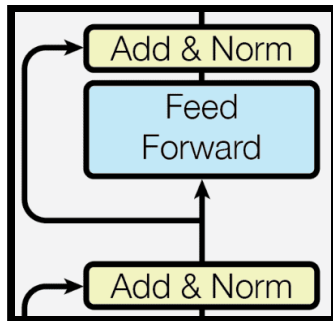
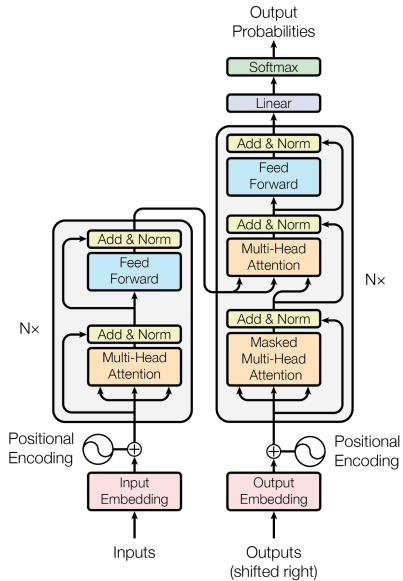
GPT-3 tiene **12.288 dimensiones** y **50.257 "palabras" (tokens)**

# Positional Encoding

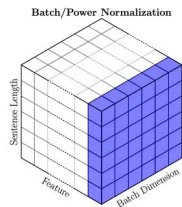
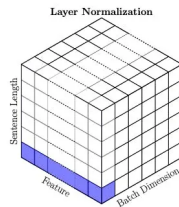
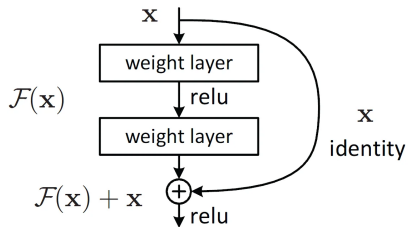


**Contexto de 2.048 tokens**

# Transformer

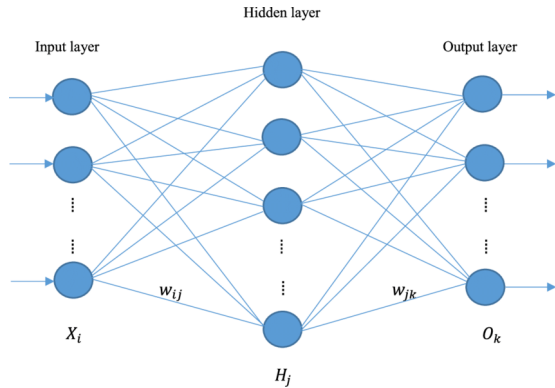
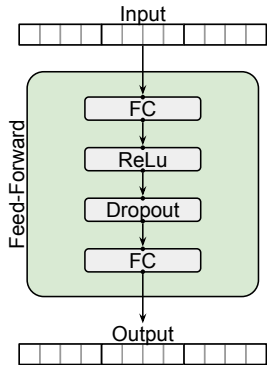


# Add & Norm

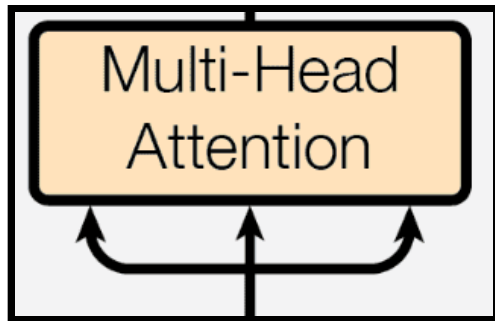
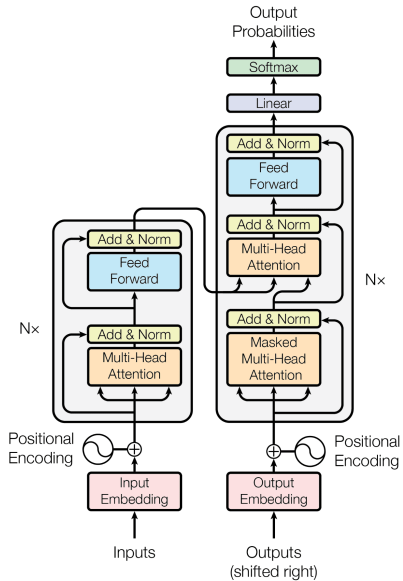




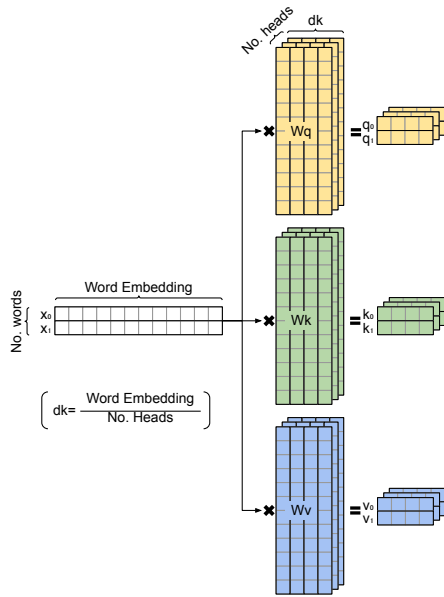
# Feed Forward



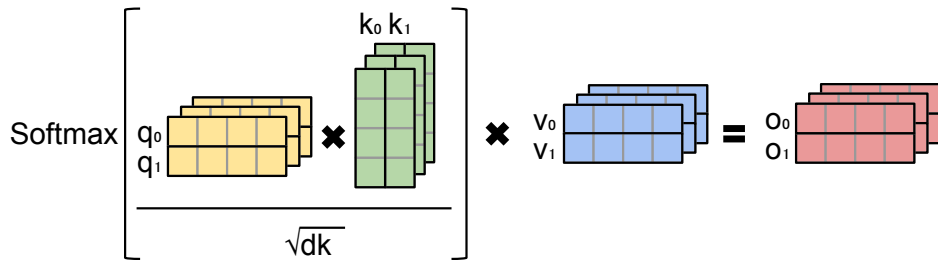
# Transformer



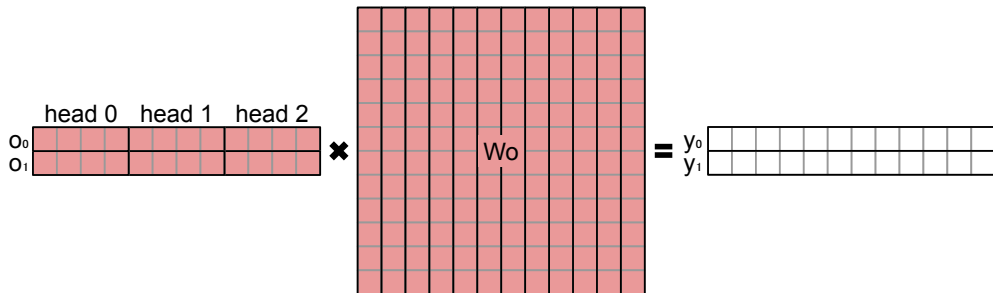
# Multi-Head Attention



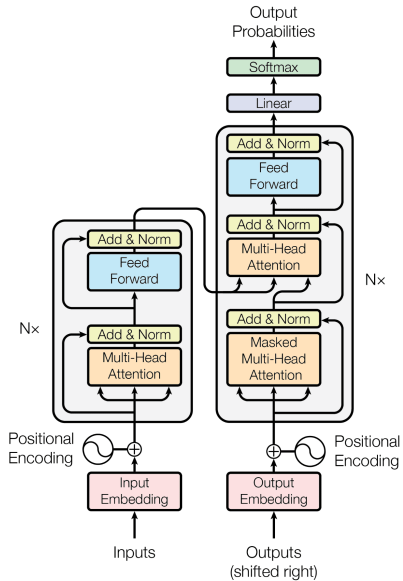
# Multi-Head Attention



# Multi-Head Attention



# Transformer



# Modelos prediseñados

<https://pytorch.org/vision/0.20/modelsclassification>

```
1 from torchvision import models
2
3 model = models.alexnet()
4 print(model)
5
6 for pretrained_weights in models.get_model_weights("alexnet"):
7     print(pretrained_weights)
8 model = models.alexnet(weights="AlexNet_Weights.IMAGENET1K_V1")
```

[https://github.com/jmiravet/curso\\_ia.git](https://github.com/jmiravet/curso_ia.git)



# Líneas de investigación

- Poda de parámetros
- Cuantización
- Fusión de capas
- Métodos para identificar y mitigar sesgos
- Interpretabilidad y explicabilidad
- Tolerancias a fallos
- Privacidad en compartición de datasets: aprendizaje federado
- Aplicaciones: Simular funciones con alto coste computacional