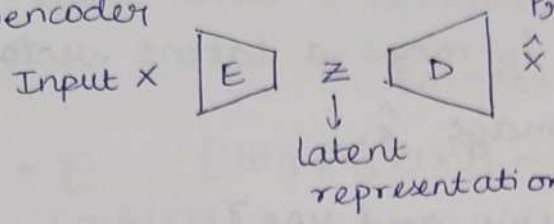
13th Jan '25
Monday.

# Computer Vision.

Autoencoder




Eg: Denoising autoencoders.
Project noisy image to latent space which is again reconstructed back to give denoised image.

Encoder E : $f_E : X \times \Theta_E \to Z \to f_E(X, \theta_E) \to Z$.

Decoder D : $f_D : Z \times \Theta_D \to X \hookrightarrow f_D(Z, \theta_D) \to \hat{X}$.

$X$ : input space
$Z$ : Latent space
$\Theta_E$ : Encoder parameter space.
$\Theta_D$ : Decoder parameter space.

$$(\theta_E, \theta_D)^* = \arg\min_{\theta_E, \theta_D \in \Theta_E \times \Theta_D} \mathcal{L}(\theta_E, \theta_D)$$ → minimize the reconstruction loss

$$= \arg\min_{\theta_E, \theta_D} \sum_{i=1}^{N_{tr}} (x^i - \hat{x}^i)^2$$

A potential candidate for $f_E$ is a CNN.
↓
composition of non-linear functions.

A potential candidate for $f_D$ is DeconvNet.

Q. Is it supervised or unsupervised?

Ans:- We generally assume it is unsupervised.
We are trying to predict the input itself.
$p(x)$ is ~~unknown~~ not known to us.

Note: The encoder $f_E$ maps an input image to a latent representation $z$.
The decoder $f_D$ maps a latent vector $z$ to a predicted image $\hat{x}$.

Q. Difference b/w this and VAE?
A. VAE would map to distribution.
We assume smoothness in the latent representation.

## Variational Auto Encoder (VAE)

- The encoder $Q(z|x)$ maps an image to a latent representation $N(\mu(x), \Sigma(x))$
- The decoder maps a latent sample $z$ drawn from $N(0, I)$ to $N(f(z, \theta_D), \sigma^2 I)$

• $N(\mu(x), \Sigma(x))$ need not be a standard normal distribution.
The encoder tries to make the KL-divergence of $N(\mu(x), \Sigma(x))$ and standard normal dist as close as possible.

• Encoder E : $f_E : X \times \Theta_E \rightarrow Z$
D : $f_D : Z \times \Theta_D \rightarrow X$

Specifically: $Q(z|x)$ is an encoder that maps $X$ to $N(\mu(x), \Sigma(x))$.
$P(x|z)$ is a decoder that maps $Z$ to $N(f(z), \sigma^2 I)$

Note that conditioning is important here.

Recall: We start with $D(\underbrace{Q(z|x)}_{\text{Encoder}} \| \underbrace{P(z|x)}_{\text{unknown}})$

$$= \mathop{\mathbb{E}}_{z \sim Q(z|x)} [\log [Q(z|x)] - \log [P(z|x)]]$$

$$= \left( P(z|x) \cdot P(x) = P(x|z)\, P(z) \right]$$

$$= D(Q(z|x) \| P(z)) - E[\log P(x|z)] + \log P(x)$$

↓ We assume that this is standard dist. That is what we assume for prior.

↓ Here there is no expectation because it doesn't depend on z.

$$\underbrace{\log P(x) - D(Q(z|x) \| P(z|x))}_{\text{unknown}} = \underbrace{E[\log P(x|z)]}_{\text{Decoder}} - D(\underbrace{Q(z|x) \| P(z)}_{\text{Encoder}})$$

$P(x|z)$ is assumed to be gaussian.

$P(z)$ is assumed to be standard normal.

Recall $P(x|z) \sim \mathcal{N}(f(z, \theta_D), \sigma^2 I)$

$Q(z|x) \sim \mathcal{N}(\mu(x), \Sigma(x))$

$P(z) \sim \mathcal{N}(0, I)$

20/01/2025
Monday

# Computer Vision

Q. How does encoder help in training?

Takes image space, maps to distribution in latent space. Ensures homogenity / smoothness in manifold. Designed in such a way that conditional distribution is similar to ~~q(z)~~ standard gaussian.

→ We want to work with tractable solution and make the optimization tractable.

→ The RHS terms: The KL divergence b/w encoder output and P(Z) is tractable.

- KL divergence b/w two standard normal dist. is 0.
- We want to bring the loss to be similar to KL divergence.

Q. Why small time steps?

Non-equilibrium thermodynamics.
Brownian motion in small time steps.
Processes are identical in both forward time and reverse time.
Both distributions have same functional form which is gaussian and loss function of both fwd and bkd would be KL divergence.

Forward process — diffusion till time step T when it would be normal dist.

→ Note that this process is controlled. Controlled amount of noise is added from timestep 0 to timestep t to generate $x_t$ which we sample at time t.

$x_t$ can be expressed in terms of $x_0$ and some noise.

$\beta$'s can either be learned or determined.

$q(x^0)$ would be distribution of data. (original).

→ We need to learn parameters of backward process such that likelihood is maximized. ie, when we draw samples from $p(x_0^0)$, they should be similar to those picked from $q(x_0)$.

These are parameters of multivariate gaussian distribution.

→ We learn how much noise has to be added at each time step.

→ Note that functional form is same for fwd and bkd but not identical.

• Markov chain : Both forward and backward trajectories are Markov chains, specifically first order.

$q(x^{(0...T)})$ is the forward distribution.

$P_\theta(x^{(0...T)})$ is the reverse distribution.

Ideally $p_\theta(x^{(0)}) \sim q(x^{(0)})$

to be learned — data dist.

→ $q$ is deterministic

$p$ has ~~to be~~ parameter $\theta$, which we are going to learn.

Goal: $\max_{\theta \in \Theta} \mathcal{L}(\theta) = \mathbb{E}_{x^{(0)} \sim q}\left[\log p_\theta(x^{(0)})\right]$ —— (1)

i.e. we want to maximize the likelihood of generating samples $p_\theta(x^{(0)})$ coming from $q(x^{(0)})$.

At time T, both forward and reverse dist. would have gaussian dist.

$$p_\theta(x^{(0)}) = \int p_\theta(x^{(0\ldots T)})\, dx^{(1\ldots T)}$$

$$\hookrightarrow \equiv p_\theta(x^{(0)}, x^{(1)}, \ldots x^{(T)})$$

$$= \int p_\theta(x^{(T)}) \prod_{t=1}^{T} p_\theta(x^{(t-1)} | x^{(t)})\, dx^{(1\ldots T)} \quad \text{—— (2)}$$

(standard result for first order Markov chains)

$$\left[ p_\theta(x^{(0\ldots T)}) = \underbrace{p_\theta(x^{(0)} | x^{(1\ldots T)})}_{\text{(due to Markov property)}}\, p_\theta(x^{(1\ldots T)}) \right.$$

$$= p_\theta(x^{(0)} | x^{(1)}) \cdot p_\theta(x^{(1\ldots T)})$$

$$\left. = p_\theta(x^{(T)}) \prod_{t=1}^{T} p_\theta(x^{(t-1)} | x^{(t)}) \right]$$

Also we assume $p_\theta(x^{(t-1)} | x^{(t)}) \sim \mathcal{N}(x^{(t-1)}; f_\mu(x^{(t)}; \theta), f_\Sigma(x^{(t)}; \theta))$ —③.

$$p_\theta(x^{(0)}) = \int p_\theta(x^{(T)}) \prod_{t=1}^{T} p_\theta(x^{(t-1)} | x^{(t)}) \cdot dx^{(1\cdots T)}$$

$$= \int p_\theta(x^{(T)}) \prod_{t=1}^{T} p_\theta(x^{(t-1)} | x^{(t)}) \cdot \frac{q(x^{(1\cdots T)} | x^{(0)})}{q(x^{(1\cdots T)} | x^{(0)})} dx^{(1\cdots T)}$$

$$= \int p_\theta(x^{(T)}) q(x^{(1\cdots T)} | x^{(0)}) \prod_{t=1}^{T} p_\theta(x^{(t-1)} | x^{(t)}) \, dx^{(1\cdots T)}$$

$$\prod_{t=1}^{T} q(x^{(t)} | x^{(t-1)})$$

Since forward trajectory is a markov chain.

21/01/2025
Tuesday

# Computer Vision

Recap:

- Diffusion Process → fwd and bwd trajectories have the same dist. function form.
- Maximizing log likelihood.
- Markov chain assumption
- Conditional dist. Gaussian → Closed form expression for bound.

• Lower bound on the expected log likelihood.

Recap:

$$\mathcal{L}(\theta) = \int q(x^{(0)}) \cdot \log P_\theta(x^{(0)}) \cdot dx^{(0)} \quad \text{—— ①}$$

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\, \mathcal{L}(\theta) \quad \text{—— ②}$$

$$P_\theta(x^{(0)}) = \int P_\theta(x^{(0\ldots T)}) \cdot dx^{(1\ldots T)}$$

$$= \int P_\theta(x^{(T)}) \prod_{t=1}^{T} P_\theta(x^{(t-1)} | x^{(t)})\, dx^{(1\ldots T)}$$

$$P_\theta(x^{(0)}) = \int P_\theta(x^{(T)}) \cdot q(x^{(1\ldots T)} | x^{(0)}) \cdot \frac{\prod_{t=1}^{T} P_\theta(x^{(t-1)} | x^{(t)})}{\prod_{t=1}^{T} q(x^{(t)} | x^{(t-1)})}\, dx^{(1\ldots T)}$$

Plug ③ into ①. —— ③

$$\mathcal{L}(\theta) = \int q(x^{(0)}) \cdot \log \left[ \int P_\theta(x^{(T)}) \cdot q(x^{(1\ldots T)} | x^{(0)}) \frac{\prod_{t=1}^{T} P_\theta(x^{(t-1)} | x^{(t)})}{\prod_{t=1}^{T} q(x^{(t)} | x^{(t-1)})}\, dx^{(1\ldots T)} \right] dx^{(0)}$$

We will apply the Jensen's inequality.

For a concave function, $Q(x)$,

$$Q(E(x)) \geq E(Q(x))$$

We pull expectation outside and log inside.

$\mathcal{L}(\theta) \geq k$, where

$$k = \int q(x^{(0)}) \cdot q(x^{(1\cdots T)}|x^{(0)}) \log P_\theta(x^{(T)}) \frac{\prod P_\theta(x^{(t-1)}|x^{(t)})}{\prod q(x^{(t)}|x^{(t-1)})} dx^{(0\cdots T)}$$

$$= \int q(x^{(0\cdots T)}) \log P_\theta(x^{(T)}) \frac{\prod P_\theta(x^{(t-1)}|x^{(t)})}{\prod q(x^{(t)}|x^{(t-1)})} dx^{(0,..T)} \quad \text{—④}$$

Since the forward trajectory is a forward chain,

$$q(x^{(t)}|x^{(t-1)}) = q(x^{(t)}|x^{(t-1)}, x^{(0)})$$

↳ ~~Since~~ This gives the flexibility of working with image. Since $x^{(0)}$ is the image R.V.

$$q(x^{(t)}|x^{(t-1)}) = q(x^{(t-1)}|x^{(t)}, x^{(0)}) \cdot \frac{q(x^{(t)}|x^{(0)})}{q(x^{(t-1)}|x^{(0)})} \quad \text{—⑤}$$

Plug ⑤ into ④.

$$k = \int q(x^{(0\cdots T)}) \cdot \log P_\theta(x^{(t)}) \frac{\prod P_\theta(x^{(t-1)}|x^{(t)})}{\prod q(x^{(t-1)}|x^{(t)}, x^{(0)})} \frac{q(x^{(t-1)}|x^{(0)})}{q(x^{(t)}|x^{(0)})} dx^{(0\cdots T)} \quad \text{—⑥}$$

Check: We want to express ⑥ as a sum of KL ~~dive~~ divergence terms and entropy terms.

$$K = \int q(x^{(0\ldots T)}) \log P_\theta(x^{(T)})\, dx^{(0\ldots T)} +$$

$$\int q(x^{(0\ldots T)}) \log \left[ \frac{\prod P_\theta(x^{(t-1)}|x^{(t)})}{\prod q(x^{(t-1)}|x^{(t)}, x^{(0)})} \cdot \frac{q(x^{(t-1)}|x^{(0)})}{q(x^{(t)}|x^{(0)})} \right] dx^{(0\ldots T)}$$

24/10/2025
Friday

## Computer Vision.

### Recap

- Diffusion Process – forward and backward trajectories.
- Markov chains

$$q(x^{(0\ldots T)}) = q(x^{(0)}) \cdot \prod_{t=1}^{T} q(x^{(t)}|x^{(t-1)})$$

$$P_\theta(x^{(0\ldots T)}) = P_\theta(x^{(T)}) \prod_{t=1}^{T} P_\theta(x^{(t-1)}|x^{(t)})$$

- Marginal :

$$P_\theta(x^{(0)}) = \int P_\theta(x^{(0\ldots T)})\, dx^{(1\ldots T)}$$

- Rewriting

$$P_\theta(x^{(0)}) = \int P_\theta(x^{(T)} \cdot q(x^{(1\ldots T)}|x^{(0)}) \cdot \frac{\prod_{t=1}^{T} P_\theta(x^{(t-1)}|x^{(t)})}{\prod_{t=1}^{T} q(x^{(t)}|x^{(t-1)})}\, dx^{(1\ldots T)}$$

Reverse

Forward

• Problem :

$$\max \mathcal{L}(\theta) = \max \int q(x^{(0)}) \cdot \log P_\theta(x^{(0)}) \cdot dx^{(0)}$$

• Lower bound :

$$\mathcal{L}(\theta) \geq K(\theta) \quad \text{(Jensen's inequality)}$$

$$K(\theta) = \int q(x^{(0\ldots T)}) \cdot \log \left[ P_\theta(x^{(T)}) \frac{\prod_{t=1}^{T} P_\theta(x^{(t-1)}|x^{(t)})}{\prod_{t=1}^{T} q(x^{(t)}|x^{(t-1)})} \right] dx^{(0\ldots T)} \quad \text{—— (1)}$$

• Using Markov property :

$$q(x^{(t)}|x^{(t-1)}) = q(x^{(t)}|x^{(t-1)}, x^{(0)})$$

$$= q(x^{(t-1)}|x^{(t)}, x^{(0)}) \cdot \frac{q(x^{(t)}|x^{(0)})}{q(x^{(t-1)}|x^{(0)})} \quad \text{—— (2)}$$

Plugging (2) into (1), we get

$$K(\theta) = \int q(x^{(0\ldots T)}) \left[ \log \left[ P_\theta(x^{(T)}) + \log \frac{\prod_{t=1}^{T} P_\theta(x^{(t-1)}|x^{(t)})}{\prod_{t=1}^{T} q(x^{(t-1)}|x^{(t)}, x^{(0)})} \cdot \frac{q(x^{(t-1)}|x^{(0)})}{q(x^{(t)}|x^{(0)})} \right] \right] dx^{(0\ldots T)} \quad \text{—— (3)}$$

Recall :

$$KL(\mathcal{N}(x; \mu_0, \Sigma_0) \,||\, \mathcal{N}(x; \mu_1, \Sigma_1)), \quad (X \in \mathbb{R}^d)$$

$$= \frac{1}{2} \left( tr(\Sigma_1^{-1}\Sigma_0) + \underbrace{(\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0)}_{\text{this term is key to optimization.}} - d + \log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right) \right) \quad \text{—— (*)}$$

$$\mathcal{N}(x, \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} (\det \Sigma)^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right]$$

- DDPM makes use of KL divergence.
  We can use constants for covariance matrix.

  ① Parameter of the statistical model.
  ② Parameter of the learning model.

- $K(\theta)$ contains average w.r.t $q(x^{(0)})$.

- $$K(\theta) = \int q(x^{(0\ldots T)}) \left[\log P_\theta(x^{(T)}) + \log \frac{\prod P_\theta(x^{(t-1)}|x^{(t)})}{\prod q(x^{(t-1)}|x^{(t)}, x^{(0)})} \cdot \frac{q(x^{(t-1)}|x^{(0)})}{q(x^{(t)}|x^{(0)})}\right] dx^{(0\ldots T)}$$

$$K(\theta) = -\sum_{t=2}^{T} \int q(x^{(0)}, x^{(t)}) \cdot KL\left(q(x^{(t-1)}|x^{(t)}, x^{(0)}) \,\|\, P_\theta(x^{(t-1)}|x^{(t)})\right) dx^{(0)} dx^{(t)}$$

$$+ H_q(x^{(T)}|x^{(0)}) - H_q(x^{(1)}|x^{(0)}) - H_P(x^{(T)}) \quad \text{— ④}$$

Note: All terms in ④ can be evaluated exactly under the following conditions:

- $q(x^{(t)}|x^{(t-1)}) \sim \mathcal{N}(x^{(t)}; \sqrt{1-\beta_t}\, x^{(t-1)}, \beta_t I)$ — ⑤

  $\beta_t$ is deterministic schedule for the forward diffusion process.

  $\beta_t$ : deterministic hyperparameter.

- $p_\theta(x^{(t-1)} | x^{(t)}) \sim \mathcal{N}(x^{(t-1)}; f_\mu(x^{(t)}, t, \theta_\mu), f_\Sigma(x^{(t)}, t, \theta_\Sigma))$ — (6)

These functions can be CNN

- $q(x^{(t-1)} | x^{(t)}, x^{(0)}) \sim \mathcal{N}(x^{(t-1)}; \tilde{\mu}(x^{(t)}, x^{(0)}), \tilde{\beta}_t I)$ — (7)

↳ Claim is that this is also gaussian.

Key term: $E_q[\| \tilde{\mu}(\,) - f_\theta(\,) \|^2]$

Empirical observation: Covariance should be deterministic.

↳ This comes from the KL divergence between two gaussians.

27/01/2025
Monday

# Computer Vision

## Recap:

- Maximizing log likelihood : $\mathcal{L}(\theta) = E_q[\log p_\theta(x^{(0)})]$ —— (1)

Backward process :

- $p_\theta(x^{(0)})$ evaluated as a marginal :

$$p_\theta(x^{(0)}) = \int p_\theta(x^{(0\ldots T)})\, dx^{(1\ldots T)} \quad \text{—— (2)}$$

- Introduce forward process :

$$p_\theta(x^{(0)}) = \int p_\theta(x^{(T)}) \cdot q(x^{(1\ldots T)} | x^{(0)}) \cdot \frac{\prod p_\theta(x^{(t-1)} | x^{(t)})}{\prod q(x^{(t)} | x^{(t-1)})}\, dx^{(1\ldots T)}$$

(due to Markov Chain Assumption). —— (3)

Plug (3) into (1) :

$$\mathcal{L}(\theta) = \int q(x^{(0)}) \left[\log \int q(x^{(1\ldots T)} | x^{(0)}) \cdot p_\theta(x^{(T)}) \cdot \frac{\prod p_\theta(x^{(t-1)} | x^{(t)})}{\prod q(x^{(t)} | x^{(t-1)})} \cdot dx^{(1\ldots T)}\right] dx^{(0)}$$

—— (4)

Lower bound $\mathcal{L}(\theta)$ in terms of $K(\theta)$ :

$$\mathcal{L}(\theta) \geq K(\theta).$$

$$K(\theta) = \int q(x^{(0\ldots T)}) \log \left[ p_\theta(x^{(T)}) \frac{\prod p_\theta(x^{(t-1)} | x^{(t)})}{\prod q(x^{(t-1)} | x^{(t)})} \cdot \frac{q(x^{(t-1)} | x^{(0)})}{q(x^{(t)} | x^{(0)})} \right] dx^{(0\ldots T)}$$

—— (5)

After a bit of work we get

$$K(\theta) = -\sum_{t=2}^{T} \int KL\left(q(x^{(t-1)} | x^{(t)}, x^{(0)}) \,\|\, p_\theta(x^{(t-1)} | x^{(t)})\right) q(x^{(0)}, x^{(t)})\, dx^{(0)} dx^{(t)}$$

$$+ H_q(x^{(T)} | x^{(0)}) - H_q(x^{(1)} | x^{(0)}) + H_p(x^{(T)})$$

—— (6)

- Specifics : $q(x^{(t)}|x^{(t-1)}) = \mathcal{N}(x^{(t)}; \sqrt{1-\beta_t}\cdot x^{(t-1)}, \beta_t I)$ — (7)

$P_\theta(x^{(t-1)}|x^{(t)}) = \mathcal{N}(x^{(t-1)}; f_\mu(x^{(t)}, t, \theta_\mu), f_\Sigma(x^{(t)}, t, \theta_\Sigma))$.

(This is a consequence of the diffusion process).

- $\beta_t$ is assumed to be deterministic.

Covariance of the backward process is deterministic and depends on the covariance of the forward process.

- Further, the variance of the backward process is assumed to be deterministic as well. i.e.,

$$f_\Sigma(x^{(t)}, t; \theta_\Sigma) = \sigma_t^2 I$$

⇒ We end up learning $\theta_\mu$.

---

- The lower bound $\cdot K(\theta) = \mathbb{E}_q\left[\sum_{t=2}^{T} L_{t-1}\right] + \mathbb{E}_q[L_T] - \mathbb{E}_q[L_0]$.

$$L_{t-1} = KL\left(q(x^{(t-1)}|x^{(t)}, x^{(0)}) \,\|\, P_\theta(x^{(t-1)}|x^{(t)})\right)$$

$$L_{t-1} = \frac{1}{2\sigma_t^2}\left\|\tilde{\mu}_t(x^{(t)}, x^{(0)}) - f_\mu(x^{(t)}, t; \theta_\mu)\right\|^2 + \underbrace{c}_{\text{constant}}$$

— (8a)

Since the forward and backward trajectories follow (7) and (8) and

$q(x^{(t-1)}|x^{(t)}, x^{(0)}) \sim \mathcal{N}(x^{(t-1)}; \tilde{\mu}_t(x^{(t)}, x^{(0)}), \tilde{\beta}_t I)$ — (9)

$P_\theta(x^{(t-1)}|x^{(t)}) \sim \mathcal{N}(x^{(t-1)}; f_\mu(x^{(t)}, t; \theta_\mu), \sigma_t^2 I)$ — (10).

$\sigma_t^2$ is related to $\beta_t$.

$\tilde{\mu}_t$ estimating → this is the mean of $x^{(t-1)} | x^{(t)}, x^{(0)}$

$\sigma_t^2$ is related to $\beta_t$.

→ From estimating mean vector to estimating noise.

28th Jan 2025
Tuesday

## Computer Vision

### Recap:

- $\min_{\theta \in \Theta} \mathbb{E}_q \left[ \| \tilde{\mu}(x^{(0)}, x^{(t)}) - f_\mu(x^{(t)}, t; \theta) \|_2^2 \right]$ — ①

  ↳ $q(x^{(t-1)} | x^{(t)}, x^{(0)}) \sim \mathcal{N}(x^{(t-1)}; \tilde{\mu}(x^{(t)}, x^{(0)}), \tilde{\beta}_t I)$

- $\tilde{\mu}(x^{(0)}, x^{(t)}) = \frac{\sqrt{\bar{\alpha}_{t-1}}}{(1-\bar{\alpha}_t)} \cdot \beta_t x^{(0)} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)} x^{(t)}$ — ②

- $p_\theta(x^{(t-1)} | x^{(t)}) \sim \mathcal{N}(x^{(t-1)}; f_\mu(x^{(t)}, t; \theta), \sigma_t^2 I)$ — ③

$q(x^{(t)} | x^{(t-1)}) \sim \mathcal{N}(x^{(t)}; \sqrt{1-\beta_t} x^{(t-1)}, \beta_t I)$ — ④

Models like UNet can be used to solve this.

Denoising model:

$q(x^{(t)} | x^{(0)}) \sim \mathcal{N}(x^{(t)}; \sqrt{\bar{\alpha}_t} x^{(0)}, (1-\bar{\alpha}) I)$ — ⑤

or $x^{(t)} = \sqrt{1-\bar{\alpha}_t}\, \sqrt{\bar{\alpha}_t}\, x^{(0)} + \sqrt{(1-\bar{\alpha}_t)}\, \epsilon \quad \epsilon \sim \mathcal{N}(0, I)$ — ⑥

where $\alpha_t = 1 - \beta_t \quad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ — ⑦.

Can we arrive at ⑤ from ④ ?


$= \sqrt{1-\beta_1}\, x^{(0)} + \sqrt{\beta_1}\, \epsilon$

Q. Show how $x^{(t)}$ relates to $x^{(0)}$?

$x^{(0)}$ can be expressed interms of $x^{(t)}$ and $\epsilon$.

$$x^{(1)} = \sqrt{\alpha_1}\, x^{(0)} + \sqrt{\beta_1}\, \epsilon$$

$$x^{(2)} = \sqrt{\alpha_2}\, x^{(1)} + \sqrt{\beta_2}\, \epsilon$$

$$= \sqrt{\alpha_2}\left(\sqrt{\alpha_1}\, x^{(0)} + \sqrt{\beta_1}\, \epsilon\right) + \sqrt{\beta_2}\, \epsilon$$

$$= \sqrt{\alpha_1\alpha_2}\, x^{(0)} + \sqrt{1-\beta_2}\sqrt{\beta_1}\, \epsilon + \sqrt{\beta_2}\, \epsilon$$

$$= \sqrt{\alpha_2}\, x^{(0)} + \left(\sqrt{\beta_1 - \beta_1\beta_2} + \sqrt{\beta_2}\right)\epsilon$$

$1 + \beta_1 + \beta_2 - \beta_1\beta_2 - 1$

$- 2\sqrt{\beta_1 + \beta_2 - \beta_1\beta_2 - 1}$

$1 - \alpha_1\alpha_2 \quad 1 - \sqrt{1-\beta_1}\sqrt{1-\beta_2}$

$1 - \sqrt{1 - \beta_1 - \beta_2 + \beta_1\beta_2}$

$\sqrt{\alpha_2}\sqrt{\beta_1}\,\epsilon + \sqrt{1-\alpha_2}\,\epsilon \quad \left(\sqrt{\alpha_2}\sqrt{1-\alpha_1} + \sqrt{1-\alpha_2}\right)\epsilon$

$$= \left(\sqrt{\alpha_2 - \alpha_1\alpha_2} + \sqrt{1-\alpha_2}\right)\epsilon$$

$$\sqrt{\alpha_2 - \alpha_1\alpha_2 + 1 - \alpha_2 + 2\sqrt{\alpha_2 - \alpha_2^2 - \alpha_1\alpha_2 + \alpha_1\alpha_2^2}}\;\epsilon.$$

$$\sqrt{1 - \alpha_1\alpha_2 + 2\sqrt{\alpha_2\alpha_1\alpha_2(\alpha_2 - 1) - \alpha_2(\alpha_2 - 1)}}$$

$$\sqrt{1 - \alpha_1\alpha_2 + 2\sqrt{(\alpha_2 - 1)\alpha_2(\alpha_2 - 1)}}$$

$$\sqrt{1 - \alpha_1\alpha_2 + 2\sqrt{\alpha_2}(1 - \alpha_2)}$$

$$= \sqrt{1 + 2\sqrt{\alpha_2} - \alpha_2(\alpha_1 + 2\sqrt{\alpha_2})}$$

$$x^{(t)} = \sqrt{\bar{\alpha}_t}\, x^{(0)} + \sqrt{1-\bar{\alpha}_t}\, \epsilon \qquad \epsilon \sim \mathcal{N}(0,1)$$

From ⑥, $$x^{(0)} = \frac{1}{\sqrt{\bar{\alpha}_t}} x^{(t)} - \frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}} \epsilon \qquad \text{——— ⑧}$$

- We will estimate $\epsilon$.

$\epsilon$ is standard gaussian.

Plug ⑧ into ②

We want everything in terms of $x^{(t)}$.

$$\sqrt{\bar{\alpha}_t} = \sqrt{\alpha_1 \cdots \alpha_t} \; ; \; \alpha_t = 1-\beta_t$$

$$\sqrt{\bar{\alpha}_t} = \sqrt{\alpha_t \bar{\alpha}_{t-1}}$$

Writing $\tilde{\mu}(x^{(t)}, x^{(0)})$ again, ②

$$\tilde{\mu}(x^{(t)}, x^{(0)}) = \frac{\sqrt{\bar{\alpha}_{t-1}}}{(1-\bar{\alpha}_t)} \beta_t \cdot x^{(0)} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)} x^{(t)}$$

From ②,

$$\tilde{\mu}(x^{(t)}, x^{(0)}) = \frac{\sqrt{\bar{\alpha}_{t-1}}}{(1-\bar{\alpha}_t)} \beta_t \left( \frac{x^{(t)}}{\sqrt{\bar{\alpha}_t}} - \frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}} \epsilon \right) +$$

$$\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)} x^{(t)}$$

$$= \frac{1}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}} \left( \beta_t + \alpha_t (1-\bar{\alpha}_{t-1}) \right) x^{(t)} - \frac{\beta_t \cdot \epsilon}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}$$

$$= \frac{1}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}} \left( 1 + \alpha_t \bar{\alpha}_{t-1} \right) x^{(t)} - \frac{\beta_t \epsilon}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}$$

$$= \frac{1}{\sqrt{\alpha_t}} \left( x^{(t)} - \frac{\beta_t \cdot \epsilon}{\sqrt{1-\bar{\alpha}_t}} \right) \qquad \text{——— ⑨}$$

From ⑨, our $f_\mu(\ )$ effectively learns to predict the noise term in ⑨. In other words we have $f_\epsilon(x^{(t)}, \epsilon, t)$.

If $f_\mu(x^{(t)}, x^{(0)}) = \frac{1}{\sqrt{\alpha_t}}\left(x^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} f_\epsilon(\ )\right)$, then

① becomes,

$$\mathbb{E}_q\left[\frac{\beta_t^2}{\alpha_t(1-\bar{\alpha}_t)} \|\epsilon - f_\epsilon(x^{(t)}, \epsilon, t)\|^2\right] \longrightarrow (10)$$

3rd Feb '2025
Monday

# Computer Vision

## Introduction to GANs

→ Difference is the maximum likelihood approach that we have used in other models.

→ Learn parameters of Discriminator and Generator.

→ Image classifier and Image generator.

→ Discriminator accepts samples that are outputs of other models and raw data ( P(x) )

→ Gradient ascent (and Gradient descent
↳ discriminator ↳ Generator.

- Implicit model
- Game Theoretic Formulation (unlike ML formulation)
- Generator vs Discriminator.

• Discriminator function:

$$D(x;\theta_D) = \begin{cases} 1, & \text{if } x \in \text{Real Sample Set } (P_{data}(x)) \\ 0, & \text{if } x \in \text{Generated sample set } (P_{model}(x)) \end{cases}$$

(~~or class~~)

• The goal of the discriminator is to discriminate (or classify) real and generated samples as accurately as possible.

$$J(\theta_D, \theta_E) = E_{x \sim P_{data}}[\log D(x;\theta_D)] +$$

$$E_{x \sim P_{model}}[\log(1 - D(x;\theta_D))] \quad \text{——} (1)$$

- Find $\theta_D^*$ that maximizes the above cost in ①.
- Let us now bring the generator into play.
- $x \sim p_{model}(x)$

  $\Rightarrow x = G(z; \theta_G)$ where $G(z; \theta_G)$ is the generator network with parameters $\theta_G$.

Let's plug ② into ①.

$$J_D(\theta_G, \theta_G) = \mathop{\mathbb{E}}_{x \sim p_{data}} [\log [D(x; \theta_D)]] + \mathop{\mathbb{E}}_{z \sim p_z} [\log (1 - D(G(z; \theta_G); \theta_D))] \quad \text{— ③}$$

- Let's now solve the problem.
- Let's define

$$J_G(\theta_D, \theta_G) = -J_D(\theta_G, \theta_D) \quad \text{(generator's loss)} \quad \text{— ④}$$

- The discriminator and the generator play a game which is defined as:

$$(\theta_G^*, \theta_D^*) = \arg \min_G \max_D V(\theta_D, \theta_G).$$

$$= \min_G \max_D \mathop{\mathbb{E}}_{x \sim p_{data}} [\log [D(x; \theta_D)]] + \mathop{\mathbb{E}}_{z \sim p_z} [\log$$

$$+ \mathop{\mathbb{E}}_{z \sim p_z} [\log [1 - D(G(z; \theta_G); \theta_D)]]$$

Initialize both $\theta_G, \theta_D$ to random values and then let the discriminator iteratively update the values.

4th Feb'25
Tuesday

# Computer Vision

Recap: $J_D(\theta_D, \theta_G) = \mathbb{E}_{x \sim P_{data}} [\log D(x; \theta_D)] + \mathbb{E}_{x \sim P_{model}} [\log(1 - D(x; \theta_D))]$

Minimization wrt generator parameters
Maximization wrt discriminator parameters.

Q. What model could be better for generator?

A. DeConv Net.

Pass Noise Distribution through a DeconvNet.

Q. What could be better for a discriminator?

A. Any image classifier.

Q. Do we have a guarantee that the generator ~~and~~ will ultimately be able to learn $P_{data}$?

A. Yes. But under certain conditions.

Q. When is the generator minimize the best discriminator model?

A. When $P_{model} = P_{data}$. Equal to $1/2$.

Q. What are the optimal solutions for $D(x; \theta_D)$ and $G(z, \theta_G)$?

Assumption: The model has high capacity.

Discriminator: $D^*$ The optimal discriminator for a given generator G is

$$D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_{model}(x)} \quad \text{——— (1).}$$

Generator : $G^*$ The optimal generator gives

$$P_{model} = P_{data} \quad \text{——— (2).}$$

$$J(\theta_D) = \int P_{data}(x) \log [D(x)]\, dx + \int P_z(z) \log [1 - D(G(z))]\, dz.$$

$$= \int P_{data}(x) \log [D(x)]\, dx + \int P_{model}(x) \log [1 - D(x)]\, dx.$$

$$= \int P_{data}(x) \log D(x) + P_{model}(x) \log (1 - D(x))\, dx.$$

This is of the form $a \log y + b \log (1-y)$.

What can you tell about 'a' and 'b' ?

Non-negative and bounded.

When $y = \frac{a}{a+b}$, it will be optimum.

$$\boxed{D_G^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_{model}(x)}} \quad \text{——— (1)}$$

Let's now find the optimal G for $D^*$

$$V(G, D^*) = C(G) = \int P_{data}(x) . \log \left[ \frac{P_{data}(x)}{P_{data}(x) + P_{model}(x)} \right]$$

$$+ P_{model}(x) . \log \left[ \frac{P_{model}(x)}{P_{data}(x) + P_{model}(x)} \right] dx.$$

Recall $JS(p\|q) = \frac{1}{2}[KL(p\|m) + KL(q\|m)]$

$$m = \frac{p+q}{2}.$$

$$C(G) = KL\left[P_{data}(x) \,\|\, \left(\frac{P_{data}(x) + P_{model}(x)}{2}\right)\right] - \log 2$$

$$+ KL\left[P_{model}(x) \,\|\, \left(\frac{P_{data}(x) + P_{model}(x)}{2}\right)\right] - \log 2.$$

$$\boxed{C(G) = 2\,JS\left(P_{data}(x) \,\|\, P_{model}(x)\right) - \log 4.} \quad \text{---(2)}$$

$\Rightarrow C(G)$ takes a min-value of $-\log 4$ when $P_{model}(x) = P_{data}(x)$, i.e., the optimal generator $G^*$ learns the data distribution exactly.

Equations (1) and (2) give answers to our questions.

10/2/2025
Monday

# Computer Vision

## Metrics for generative models

→ We want to know the quality of images generated by models such as GAN.

### Inception Score :

How to calculate ?

There is a model Inception V3, takes as input the images we get from GANs (sample them). Calculate score based on this output.

$$IS = \exp\left[ \mathbb{E}_{x \sim P_{model}} \left[ KL\left(P(y|x) \,\|\, P(y)\right) \right] \right]$$

$P(y|x)$ → sharpness or faithfulness

$P(y)$ → diversity.

$P(y)$ → R.V y must have high entropy.

diversity

(We want outputs of model to be as diverse as possible, means distributed over all labels).

$P(y|x)$ → R.V $P(y|x)$ must have low entropy.

sharpness or faithfulness

(We want the model to output single output label for given x).

It tells how sharp the image is.

→ We want KL divergence high, means distance between them is high, so that we get images that are diverse across all labels and each image has unique label.

In general,

① How realistic is the image? (depends on classifier performance).

② Generate samples from variety of classes.

- Ideally $p(y|x)$ should be far from a uniform distribution.
- $p(y)$ must be close to uniform distribution.
- $p(y|x)$ is found from the output of the classifier (Inception V3).

$$p(y) = \int p(y|x) \cdot p(x)\, dx \quad \text{(marginal)}$$

We can approximate it by samples.

$$p(y) \approx \frac{1}{N} \sum_{i=1}^{N} p(y|x_i) \quad \text{(Approximation)}$$

Drawbacks:

We can't ~~train~~ expect ~~perfect~~ accurate results when data like medical data is used.

Why? ~~Biased~~ Becoz biased towards ImageNet like data.

Can't be reliable for non-ImageNet data.

# Frechet Inception Distance (FID):

→ You don't take single image. Sample multiple images.

→ Compare dist of generated samples with dist of real samples.

→ Compare mean and covariance of real samples with that of generated ones.

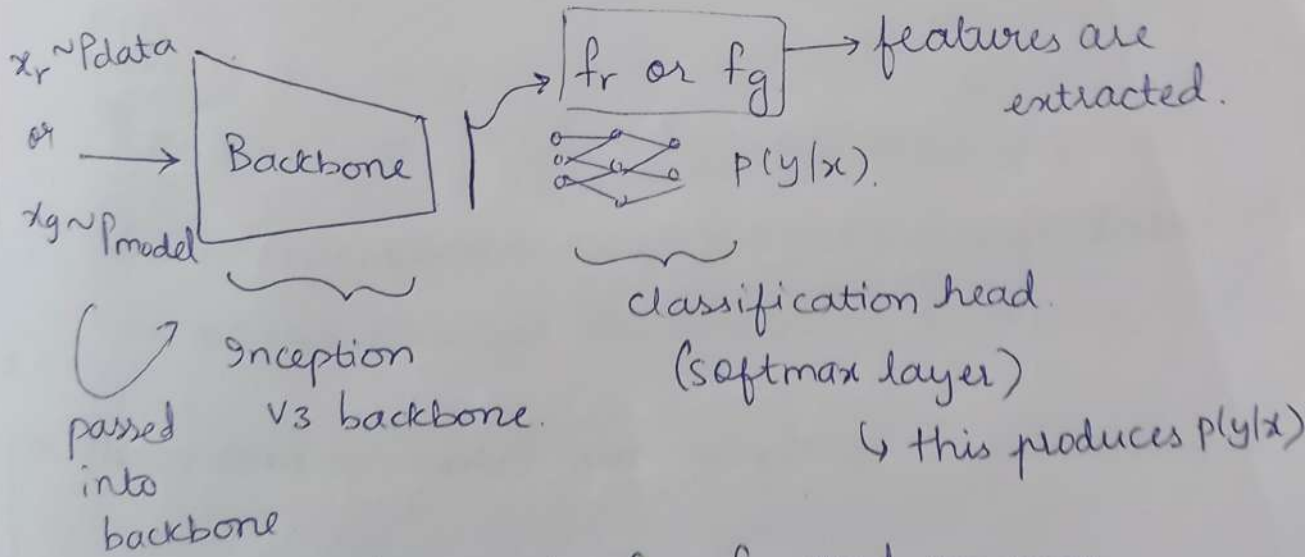
$$FID = \| \mu_r - \mu_g \|_2^2 + tr\left( \Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2} \right)$$

→ We want this value to be low.

$\mu_r$ : Mean vector of the Inception vector of real samples.

$\mu_g$ : Mean vector of the Inception vector of generated samples.

$\Sigma_r, \Sigma_g$ : covariance matrices of real and generated samples respectively.




We want $f_r$, $f_g$ and compare distributions.

The more the samples, the better.

Ideally, a generative model evaluation metric must:

1. Measure sharpness
2. Measure diversity
3. Measure "realness" / "naturalness".

Naturalness Image Quality Evaluator.

11/02/2025
Tuesday

# Computer Vision

## Recap:

1. How real the image is?
2. How diverse labels are?

## Drawbacks of Inception Score:

1) Non-ImageNet Type data has problems.
   Why this could be issue?
   See cond dist $p(y|x)$ of model.
   If y and x come from data it hasn't learnt, then it could be problem.

2) Doesn't look for mode collapse.

> Here $p(y|x)$ has high entropy and $p(x)$ has less entropy

Mode collapse means model generates not so diverse images.
This model can fool classifier and give high IS
How? Models have few variations in same image.

## FID

$$\|\mu_r - \mu_g\|_2^2 + tr\left(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}\right)$$

↓
Mean representation we get when real data is passed through the inception backbone.

→ Downstream model we considered is Inception V3.

# Naturalness Image Quality Estimator (NIQE)

$I(i,j)$ → Natural image grayscal (captured from camera)

↓

pixel intensity at spatial location $(i,j)$.

$I(i,j) \xrightarrow{\text{MSCN}} \hat{I}(i,j)$

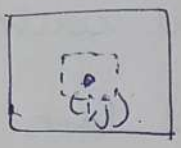
Mean subtracted contrast Normalization

This is a transformation.

$$\hat{I}(i,j) = \frac{I(i,j) - \mu(i,j)}{\sigma(i,j) + C}$$

$\mu(i,j)$ → local mean around that square.

$\sigma(i,j)$ → local S.D.




Why $C$ ?

To prevent division with '0'.

$$\mu(i,j) = \frac{1}{N} \sum\sum_{m,n \in N(i,j)} I(i-m, j-n)$$

↓

Neighbourhood.

$$\sigma^2(i,j) = \frac{1}{N} \sum\sum_{m,n \in N(i,j)} \left(I(i-m, j-n) - \mu(i,j)\right)^2$$

↓

This is called contrast.

Difference between $I(i,j)$ and $\mu(i,j)$ locally.

→ In natural images, the histogram (~~dist~~ (distribution of $\hat{I}$) turns out to be gaussian. It is unimodal (single peak).

So, for natural images, we have more handle on the statistical patterns of the normalized $\hat{I}$.

This can be used as a signature to characterize natural images.

→ If we take synthetic images (generated from models), then the histogram is multi-modal (many peaks).

Unimodal is true only in case of natural images.

→ $\hat{I}$ is unimodal, and can be modelled using either a Gaussian or a Generalized Gaussian Distribution (GED).

Q. Can we characterize all natural images?

FID score for real images vs synthetic images.

MSCN Histogram

→ When blurring (remove many details), so pixel values become more similar to the local ~~average~~ mean

So histogram becomes narrower because there is less variation.

→ When we add noise, pixel I become random. Std deviation increases. So histogram becomes wider. Spread in variance of image.

→ If we characterize the statistics of real undistorted images (Gaussian), we can build a model so that when new image comes we can detect whether it is fake or not based on these ~~the~~ statistics.

→ $(\alpha, \beta)$ uniquely identify gaussian.

↳ shape of GGD (Generalized Gauss. Dist.)

Natural images have specific range of $\alpha, \beta$.

So we can train a model to check if new image's $(\alpha, \beta)$ fall into that range.

→ $\gamma$ is shape parameter (~~"small~~ tells (sharp or flat) ~~the~~ distribution

$\beta_l, \beta_r$ (left and right tails of dist).

↳ tells how extreme pixel values behave.

14/02/2025
Friday

# Computer Vision

Image Net has fixed no of classes.

Where Consequence: Generalization

Observation: Language guidance will help.

→ ResNet, VIT: extract ~~fea~~ image features.

Transformers: Extract language (text)
(BERT, GPT)

→ Model should work like, take as input (image, text) pair and give latent Representation. Maps both to common latent space.

## Contrastive language Image Pretraining (CLIP)

→ NO finetuning is required. It learns generalizable representation.

→ Zero-shot learning: No need of training data for new things.
↳ DIDN'T UNDERSTAND

→ Fixed set of categories is limiting.

→ Learning from raw text about images is promising.

→ Pretraining with (text, image) pairs.

→ Info Noise Contrastive Estimation (info NCE) loss.
Pulls similar pairs closer in latent spaces and pushes unrelated ones far apart.

→ Enables zero-shot ~~learning~~ transfer to downstream

tasks.

→ Natural language based.

→ Cosine similarity + softmax.

cosine similarity → closeness

softmax → predict most relevant ones

**Info NCE loss:**

$$L = -\sum_{i=1}^{N} \log \left( \frac{\exp(S(I_i, T_i)/\tau)}{\sum_{j=1}^{N} \exp(S(I_i, T_j)/\tau)} \right)$$

$\tau$ → temperature parameter. controls sharpness of dist.

softmax

where $I_i$ is the $i^{th}$ image representation and $T_i$ is the $i^{th}$ caption representation

$$S(x, y) = \frac{\langle x, y \rangle}{\|x\| \cdot \|y\|}$$

17th Feb '25
Monday

# Computer Vision

→ Contrastive pretraining plays major role in unsupervised learning.

CLIP helps in:

1. Zero-shot learning : No need for fine-tuning
2. Better generalization.
3. Learned features can be used in downstream tasks.

Q. Why same dimension for image and text?

- Cosine similarity.
- Uniform Representation space.

- Info NCE loss.
  - Representation learning in unsupervised learning.
  - The role of mutual Information (MI).
    - In contrastive learning, positive pairs ⇧ higher MI.
    - ⇧ Info NCE optimizes MI (minimize negative pair similarity).
  - Info NCE Loss Formulation and Negative Sampling.
  - Relation to MI.

Info NCE loss:

$$\mathcal{L} = -\sum_{i=1}^{N} \log \left[ \frac{\exp(S(I_i, T_i)/\tau)}{\sum_{j=1}^{N} \exp(S(I_i, T_j)/\tau)} \right]$$

(positive)

Here we took $L_{image}$ as reference.

Similarly we can have $\mathcal{L}_{Text}$ as reference.

What is Info NCE ?

Info → Noise
NCE → Contrastive

Contrasts positive and negative samples.

Q. Can we do prediction in latent space?

A. Yes. Zero-shot transfer allowed to new tasks without additional training.

→ Auto encoder: Deterministic setting.
Maps input data to latent rep. deterministically.

→ Variational Auto encoder: Stochastic setting.

→ CLIP solves a semisupervised setting to learn representations.
Doesn't require labelled data. Can learn from pairs.

→ Goal: Learn latent representations.
Advantage: ① Can work with lower dimension.
② Better generalization.
③ Useful for downstream tasks.

→ Optimizing Info NCE loss is equivalent to maximizing a lowerbound for M.I.

→ We want Info NCE loss to be low.

## Predictive Coding in Latent Space.

→ Helps to model dependencies between past and future representations.

Goal: Maximize Mutual information (MI) between the source (input) and the context (history).

→ If MI is high, context helps in predicting future.

→ We use two-model in Predictive Coding:

① Learn latent representation from input sample (Eg: Speech).

② Auto-regressive model for History.
Remembers the past context and predicts future representations by maximizing MI b/w past and future latent.

Q. What is Mutual Information (M.I)?

If X, Y are R.V, M.I(X, Y) tells how dependent is one R.V on another.

We quantify this using entropy.

$$MI(X, Y) = H(X) - H(X|Y)$$

Entropy of X: No. of bits to represent X without knowing Y.

If X, Y are independent $MI = 0$.

Mutual Information (M.I):

$$I(X:Y) = KL(p(x,y) \,\|\, p(x) \cdot p(y)) \rightarrow p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

$$p(x,y) \log \frac{p(x|y)p(y)}{p(x)p(y)}$$

$$= \underset{(x,y) \sim p(x,y)}{E} \left[ \log \left( \frac{p(x|y)}{p(x)} \right) \right]$$

$$= \int p(x,y) \cdot \log \left[ \frac{p(x|y)}{p(x)} \right] dx \, dy.$$

→ We want $z_{t+k}$ (latent representation) at time 't+k' such that M.I between the original speech input $x_{t+k}$ and context representation ($c_t$) is maximized.

↓

Summarizes past information.

→ In CPC, we want to learn the latent representation $z_{t+k}$ at time 't+k' such that $I(x_{t+k}; c_t)$ is maximized.

- Find a function $f_k(x_{t+k}, c_t)$ such that $I(x_{t+k}, c_t)$ is maximized.

  Let $f_k(x_{t+k}; c_t) = \exp(z_{t+k}^T \cdot W_k \cdot c_t)$ be one candidate function.

18/02/2025
Tuesday

# Computer Vision

Auto regressive model helps in looking at long back history and not just the local spatial window.

Applications with the encoder we have learnt:
helps in classifying.

Applications from CLIP: Zero-shot learning.

M.I: $I(X:Y) = KL(P(x,y) \| p(x).p(y))$

$$I(x;y) = \int p(x,y).\log \frac{P(x|y)}{P(x)}\, dx.dy$$

We want to learn $Z_{t+k}$ such that $I(x_{t+k}; c_t)$ is maximized.

$Z_{t+k}$ → d.r at future time instant.

$$I(x_{t+k}; c_t) = \sum_{x_{t+k}, c_t} p(x_{t+k}, c_t) \log \frac{P(x_{t+k}|c_t)}{P(x_{t+k})}$$

we are interested in this term.

- Specifically can we learn a function $f_k(x_{t+k}, c_t)$ that is proportional to $\frac{P(x_{t+k}|c_t)}{P(x_{t+k})}$

What is that range of this?
$[0, \infty)$.

- Note that $\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$ is non-negative.

→ Encoder helps with latent rep.
CNN can be used for images and
Transformer can be used for text/speech.

- A candidate function for approximating $\frac{P(x_{t+k}|c_t)}{P(x_{t+k})}$ is $f_k(x_{t+k}, c_t) = \exp(z_{t+k}^T \cdot W_k \cdot c_t)$

This func. is non-negative

- Info NCE loss func. is

$$\mathcal{L} = -\frac{1}{N}\sum_{k=1}^{N} \log\left(\frac{f_k}{\sum_{j=1}^{N} f_j}\right)$$

=

19th Feb '25
Wednesday

# Computer Vision

Recap:

$$L = -E_X \log \left( \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_j(x_j, c_t)} \right) \quad \text{---(1)}$$

$-E_X$: average

- We want to maximally preserve $I(x_{t+k}; c_t)$ ($x_{t+k}$: raw input, $c_t$: context)

$$= \sum_{x_{t+k}, c_t} p(x_{t+k}, c_t) \log \left( \frac{p(x_{t+k} | c_t)}{p(x_{t+k})} \right)$$

- $f_k(x_{t+k}, c_t) = \exp\left( z_{t+k}^T W_k \cdot c_t \right)$ → a candidate model we use.
  - $z_{t+k}$: encoder
  - $c_t$: autoregressive model.

to predict $\boxed{\frac{p(x_{t+k} | c_t)}{p(x_{t+k})}}$ → This comes from MI preservation condition.

→ We want to learn the encoder and autoregressive model together.

- For us to jointly learn the encoder and the autoregressive model, we use the info NCE loss in (1)

$$\mathcal{L} = -E_X \left[ \log \left[ \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_j(x_j, c_t)} \right] \right]$$

numerator: positive pair

denominator: all pairs of raw inputs and $c_t$.

$X$ is the set of data samples, i.e.,

$$X = \{x_1, x_2, \ldots, x_N\}.$$

In this set there is one positive sample (correct one) and $N-1$ negative samples.

- Let's assume that we find our optimal encoder and the auto regressor.

$$f_k(x_{t+k}, c_t) = \frac{p(x_{t+k}|c_t)}{p(x_{t+k})} \quad \text{——— (4)}$$

Recap:

$$I(x_{t+k}; c_t) = \sum_{(x_{t+k}, c_t)} p(x_{t+k}, c_t) \cdot \log \frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$$

Plug (4) into (3)

$$\mathcal{L}_{opt} = -E_x \log \left[ \frac{\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}}{\frac{p(x_{t+k}|c_t)}{p(x_{t+k})} + \sum_{j \neq t+k} \frac{p(x_j|c_t)}{p(x_j)}} \right]$$

$$\overset{**}{\approx} E_x \log \left[ 1 + \frac{p(x_{t+k})}{p(x_{t+k}|c_t)} \cdot (N-1) E_{x_j} \frac{p(x_j|c_t)}{p(x_j)} \right]$$

$$\left( E_{x_j} \frac{p(x_j|c_t)}{p(x_j)} = \int p(x_j) \cdot \frac{p(x_j|c_t)}{p(x_j)} \cdot dx_j \right)$$

$$= 1.$$

$$\overset{*}{=} E_x \log \left[ 1 + \frac{p(x_{t+k})}{p(x_{t+k}|c_t)} \cdot (N-1) \right]$$

$$\geq E_x \log \left[ N \cdot \frac{p(x_{t+k})}{p(x_{t+k}|c_t)} \right]$$

$$= \log N - E_x \log \left[ \frac{p(x_{t+k}|c_t)}{p(x_{t+k})} \right] \leftarrow I(x_{t+k}; c_t)$$

$$\mathcal{L}_{opt} \geq \log N - I(x_{t+k}; c_t)$$

$$\boxed{I(x_{t+k}; c_t) \geq \log N - \mathcal{L}_{opt}} \quad \text{——— (6)}$$

24th Feb 2025
Monday

# Computer Vision

## Recap:

Downstream task of CLIP (Pretraining) is zero-shot transfer.

- Contrastive learning helps to leverage large scale dataset without explicit annotation or labelling.
- CLIP is an example model.
- Contrastive learning : pulls +ve pairs together.
  pushes -ve pairs apart.
- Info NCE loss is a "good" function that encourages contrastive learning.

## Noise-Contrastive Estimation (NCE)

We have $X = \{x_1, x_2, \ldots, x_N\}$ drawn from unknown data distribution. $P_{data}(\cdot)$ (unknown).
A dataset of N points.

→ We assume $P_{data}(\cdot) \in \{P_{model}(\cdot\,; \alpha)\}_\alpha$
(family of priors).

→ For some optimal $\hat{\alpha}$, we have,

$\exists\, \hat{\alpha}$, such that $P_{data}(\cdot) = P_{model}(\cdot\,; \hat{\alpha})$.

and $\underbrace{\int P(x; \hat{\alpha})\, dx = 1}$

constraint.

→ Any model,

$$P_{model}(\cdot;\alpha) = \frac{1}{Z(\alpha)} P^{\circ}_{model}(\cdot;\alpha)$$

where $P^{\circ}_{model}(\cdot;\alpha)$ is an unnormalized function.

$Z(\alpha)$ is the normalization constant.

$$Z(\alpha) = \int P^{\circ}_{model}(x;\alpha) \cdot dx. \quad \text{——— (1).}$$

→ Finding $Z(\alpha)$ in practice is usually intractable analytically.

becoz it is integral over all possible values of $x$, computationally expensive or intractable.

→ NCE solves the problem by:

NCE looks at this as a classification problem.

Compares it against a known noise distribution.

It learns $Z(\alpha)$ as a parameter.

It basically differentiates b/w real samples $\sim P_{data}(x)$ and noise samples $\sim P_{Noise}(x)$ and in the process of doing so, finds $P^{\circ}_{model}(\cdot;\alpha)$.

— using $P^{\circ}_{model}(\cdot;\alpha)$

— making $Z(\alpha) \approx C$ a learnable parameter.

- NCE learns $\theta = \{\alpha, c\}$ in a "noise contrastive" approach.
- In other words, we learn the data dist. in a relative or contrastive fashion w.r.t known noise dist. $P_z(\cdot)$.

25/02/25

# Computer Vision

- Recap: NCE
- NCE loss
- Connection to supervised learning.

- Recap:
  - $X = (x_1, x_2 \dots x_N)$ drawn from $p_{data}(\cdot)$.
  - $p_{data}(\cdot) \in \{ p_{model}(\cdot\,; \alpha) \}_\alpha$.
  - Find $\alpha$ that "best fits" the data points
  - We are interested in solving the problem of estimating unnormalized functions $\overbrace{p^0_{model}(\cdot\,; \alpha)}$ due to tractability issues
  - specifically finding $Z(\alpha) = \int p^0_{model}(x; \alpha)\, dx$ is difficult.
  - If $p^0_{model}(x; \alpha) \sim \mathcal{N}(0, \sigma^2)$, what is $Z(\sigma)$? $Z(\alpha) = \sqrt{2\pi}\sigma$
  - In general, find $Z(\alpha)$ is non-trivial.
  - NCE proposes a way to estimate both $\alpha$ and $\underbrace{Z(\alpha)}_{c}$ by solving a single optimization.
  - NCE accomplishes this by estimating $\theta = \{\alpha, c\}$ by comparing or contrasting it with an artificial noise distribution from which we draw N samples $Y = (y_1, y_2 \dots y_N)$.
  - Noise-contrastive estimation works by solving the following optimization. — (1)

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, J(\theta), \text{ where}$$

$$J(\theta) = \frac{1}{2N} \sum_{i=1}^{N} \log[h(x_i; \theta)] + \log[1 - h(y_i; \theta)] \quad — (2)$$

$$h(x; \theta) = \frac{1}{1 + \exp(-G(x; \theta))}\,; \quad G(x; \theta) = \ln(p'_{model}(x; \theta)) - \ln(p_n(x))$$

$$\Rightarrow h(x;\theta) = \frac{1}{1+\exp\left(\ln \frac{p_n(x)}{p^0_{model}(x;\theta)}\right)}$$

$$h(x;\theta) = \frac{p^0_{model}(x;\theta)}{p^0_{model}(x;\theta) + p_n(x)}$$

Note: $\ln\left(p_{model}(\cdot;\alpha)\right) = \ln c + \ln\left(p^0_{model}(\cdot;\alpha)\right)$

## Connection to supervised learning

Let $u = (u_1, u_2 \cdots u_{2N})$ be the union of $X$ and $Y$.

If $u_i \in X$, $c_i = 1$ Note $p(c=1) = p(c=0) = \frac{1}{2}$.

$u_i \in Y$, $c_i = 0$.

- $p(u|c=1;\theta) = p_{data}(u;\theta)$ or $p^0_{model}(u;\theta)$
- $p(u|c=0) = p_n(u)$.

What is $p(c=1|u;\theta) = \frac{p(u|c=1;\theta)\cdot p(c=1)}{p(u)}$

$$= \frac{\frac{1}{2}\cdot p^0_{model}(u;\theta)}{\frac{1}{2}\left(p^0_{model}(u;\theta) + p_n(u)\right)}$$

$$= h(u;\theta)$$

$$p(c=0|u;\theta) = 1 - h(u;\theta)$$

Now apply the log loss to these estimates.

$$l(\theta) = \sum_{i=1}^{2N} (\quad)\log\left(h(u_i;\theta)\right) + \quad -(3)$$

$$(\quad)\log\left(1-h(u_i;\theta)\right).$$

28/02/25

# Computer Vision

- Recap: NCE loss formulation
- Wrap up contrastive learning discussion

Reminders:
- PPR due tonight
- Midterm exam coming up.

- SimCLR

Setup

- $X = (x_1, x_2 \dots x_N)$ drawn from $p_{data}(x)$.
- $p_{data}(x) \in \{ p(x;\alpha) \}_\alpha$, a prior family.
- $p_{model}(x;\alpha) = \frac{1}{Z(\alpha)} \cdot p^{\circ}_{model}(x;\alpha)$
- We want to learn $Z(\alpha)$, $\alpha$ . ($\because Z(\alpha) = \int p^{\circ}_{model}(x;\alpha)\, d\alpha$ is intractable)
- We learn $\theta = \{\alpha, c\}$ in a noise contrastive fashion.
- $Y = (y_1, y_2 \dots y_N)$ drawn from $p_n(y)$.
- $J(\theta) = \frac{1}{2N} \sum_i \log [h(x_i;\theta)] + \log [1 - h(y_i;\theta)]$

$$h(x_i;\theta) = \frac{p_{model}(x_i;\theta)}{p_{model}(x_i;\theta) + p_n(x_i)}$$

$$h(x;\theta) = \frac{1}{1 + \exp(-G(x;\theta))},$$

$$\boxed{G(x;\theta) = \ln p_{model}(x;\theta) - \ln p_n(x).} \quad - (x)$$

- Recall connection with supervised learning.

$$l(\theta) = \sum_i c_i \log\left(\underbrace{p(c_i=1 \mid u_i;\theta)}_{h(x_i;\theta)}\right) + (1-c_i)\log\left(p(c_i=0 \mid u_i;\theta)\right)$$

Class labelling:
$c_i = 1$ if $u_i \in X$
$c_i = 0$ if $u_i \in Y$

$$l(\theta) = \sum_i \log h(u_i;\theta) + \log\left(1 - h(u_i;\theta)\right)$$

$$U = X \cup Y$$

- Please read the InfoNCE and NCE papers.

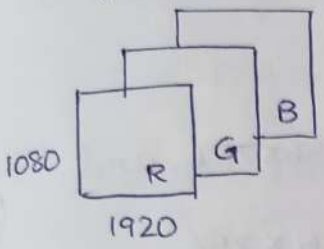
Hint:

- Go back to the InfoNCE formulation and find $p([d=i] \mid x, c_i)$ where $[d=i]$ is the event corresponding to the positive pair.

3/03/2025
Monday

# Computer Vision

How a HD image is represented in memory?
(After encoding)




1920 × 1080 × 3.
w h c

- There are 256 intensity level values. (0 to 255)
  1 byte of data per pixel.

- I(w,h,c) is a pixel intensity corresponding to spatial location (w,h) and channel c.

- Typically, 8 bits are used to represent a ~~pixel~~ pixel's intensity for each colour channel.
  ⇒ Since 3 channels, 24 bits per pixel.

- Raw RGB frames:
  1920 × 1080 × 3 bytes ≈ 6.22 MB per frame.

- ~~Images~~ Further, images (and videos) are compressed before storage.

4/03/2025
Tuesday

# Computer Vision

Recap: Image formation

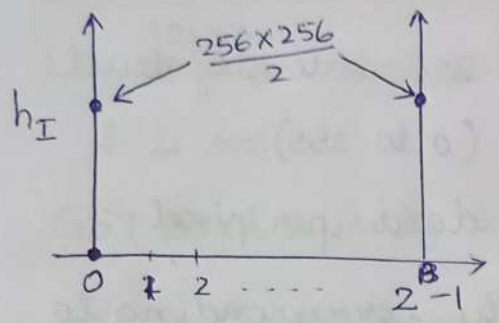
- Image histogram
- Point operations
  - linear
  - Non linear.

- **Image histogram:**

Given a gray scale image of size M×N that uses 8 bits per pixel I, the histogram of I

denoted $h_I$ is the count of the intensity levels in I, i.e.

$h_I(k)$ denotes the count of intensity value k occurring in the image.




If I were 256x256. and

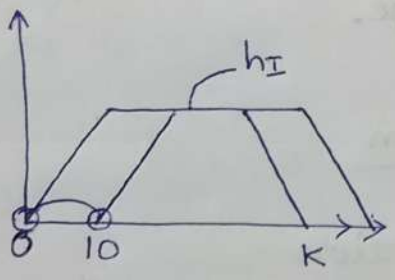
$$h_I(0) = \frac{256 \times 256}{2}$$

$$h_I(255) = \frac{256 \times 256}{2}$$

$$\sum_{k=0}^{2^B-1} h_I(k) = 256 \times 256 \text{ or generally } M \times N.$$

- $P_I(k) = \frac{h_I(k)}{M \times N}$ is the normalized histogram of I. It is a valid probability mass function. (PMF).

- **Point operations :**




Ex: $J = I + k_0 = 10$.

$h_J(k) = h_I(k - k_0)$.

In general $J = PI + L$ (linear).

J: Transformed pixel intensity
P: contrast (scaling factor)
L: offset (brightness or shift)

If $A = \min(I) > 0$

$C = \max(I) < 2^B - 1$

B = bits per channel.

Now we want to stretch the range to $[0, 2^B-1]$. This is contrast stretching.

We solve for P and L:

$$0 = P \cdot A + L$$

$$2^B - 1 = P \cdot C + L$$

- Non-linear operations :-
  - $J = (I)^r$
  - $J = \log[1+I]$ (log compression)
  - $J = \exp[f(I)]$

7/03/2025
Friday

## Computer Vision

Recap : Histogram, point operations

- LTI Systems (Linear Time Invariant)
  - Convolution Sum.
  - Low pass filter
  - High pass filter
  - Image denoising

- 'b' bit image means $2^b$ intensity levels can be supported.

### Linear filters:

→ Operations where output pixel is a linear combination of its neighbouring input pixels.

→ We apply the same operation at every pixel.

→ We used this in CNN and attention.

- A system $T\{\}$ is a map from an input sequence $x[n]$ to a desired output sequence $y[n]$.

$$y[n] = T\{x[n]\}.$$

- Linear system:

A system $T\{\}$ is said to be linear if

$$y_1[n] = T\{x_1[n]\}, \quad y_2[n] = T\{x_2[n]\}$$

$$\Rightarrow T\{ax_1[n] + bx_2[n]\} = ay_1[n] + by_2[n].$$

Desired output sequence would be to smooth out the higher frequencies.

- Time invariant:

A system $T\{\}$ is said to be time invariant if $y[n] = T\{x[n]\} \Rightarrow T\{x[n-n_0]\} = y[n-n_0]$

(Basically, time-shifted input produces time-shifted output)

- Linear & Time Invariant (TI) System:

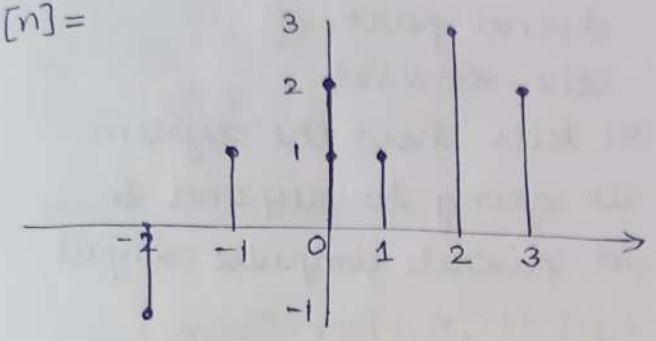
A system $T\{\}$ that satisfies both the linearity and time invariance properties simultaneously.

→ A consequence of LTI is the convolution sum.

→ Any discrete (digital) signal or sequence $x[n]$ can be expressed as a sum of scaled and shifted discrete delta functions.

$$\delta[n] = 1 \quad \text{if } n=0$$
$$0, \quad \text{if } n \neq 0.$$

$x[n] =$




$x[n] =$

| | |
|---|---|
| -1 | $n=-2$ |
| 1 | $n=-1$ |
| 2 | $n=0$ |
| 1 | $n=1$ |
| 3 | $n=2$ |
| 2 | $n=3$ |

$x[n]$ can be written as,

$$x[n] = -1.\,\delta[n-(-2)] + 1.\delta[n-(-1)] + 2.\delta[n-0] + 1.\delta[n-1] + 3.\delta[n-2] + 2.\delta[n-3]$$

- In general, any discrete signal $x[n]$ can be written as

$$x[n] = \sum_{k=-\infty}^{\infty} x[k].\,\delta[n-k].$$

→ Let's pass this signal through an LTI system. $T\{\}$.

$$y[n] = T\{x[n]\}$$
$$= T\left\{\sum_{k=-\infty}^{\infty} x[k]\,\delta[n-k]\right\}$$
$$= \sum_{k=-\infty}^{\infty} x[k]\,T\{\delta[n-k]\} \quad \text{(due to linearity)}$$

If $h[n] = T\{\delta[n]\}$

$$\Rightarrow T\{\delta[n-k]\} = h[n-k] \quad \text{(due to time invariance)}$$

$h[n]$ is called the impulse response of the system $T\{\}$.

$$\therefore y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] \longrightarrow \text{(Convolution sum)}$$

kernel part of the convNet.

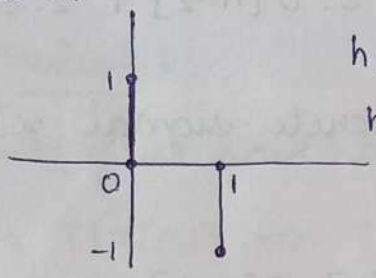
It tells how the system is going to respond to a discrete impulse input.

Example :

Let $h_1[n] = \underline{\delta[n] - \delta[n-1]}$

1

This is called as discrete differentiation operation.



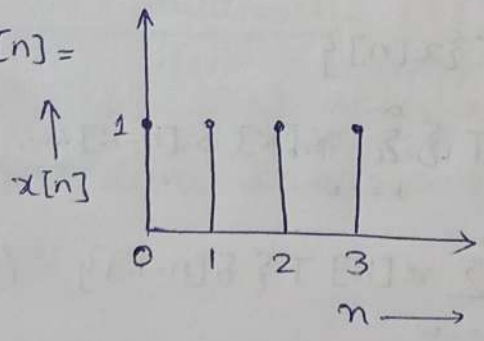

$h_1[0] = 1$

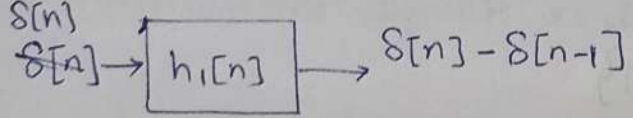
$h_1[1] = -1$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

2D version of this is used to detect edges.

Let $x[n] =$




$$x[n] = \sum_{k=0}^{3} 1 \cdot \delta[n-k].$$

$$y[0] = \sum_{k=0}^{3} x[k]\, h[-k]$$

$$= x[0]h[0] + x[1]h[-1] + x[2]\cdot h[-2] + x[3]h[-3]$$

$$= 1.1$$

$$= 1$$

$$y[2] = \sum_{k=0}^{3} x[k]\, h[2-k]$$

$$= x[0]h[2] + x[1]h[1] + x[2]h[0] + x[3]h[-1]$$

$$= 1(-1) + 1(1)$$

$$= 0$$

$$y[4] = \sum_{k=0}^{3} x[k]\cdot h[4-k] = -1$$

13th March '25
Monday

## Computer Vision

- Recap: LTI Systems, Convolution Sum.
- Linear Image Filtering → Used to emphasize or suppress certain features.
  - Low Pass
  - High Pass
- LTI systems:

Uniquely specified by its impulse response.

$$h[n] = T\{\delta[n]\}$$

- For any input $x[n] = \sum_{k=-\infty}^{\infty} x[k]\cdot \delta[n-k]$

$$y[n] = T\{x[n]\} = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k] \quad \text{——} \circledast$$

## Low Pass filters

→ Smooth the image
→ Remove noise
→ Blur sharp details like edges.

## High Pass filters

→ Edges it can detect.
→ Sharp changes features can be extracted.

- Example: $h[n] = \delta[n] - \delta[n-1]$

This is high-pass filter. Why?

A: Becoz if sharp changes are there, $x[n]$, $x[n-1]$ very different, $h[n]$ will be large, if ~~sha~~ smooth changes then $x[n] \approx x[n-1]$ ∴ so 0.

$$x[n] = \begin{cases} 0 & \text{else} \\ 1 & 0 \le n \le 3. \end{cases}$$

We ~~have~~ get

$$y[n] = \begin{cases} -1 & n = 4 \\ 0 & 1 \le n \le 3 \\ 1 & n = 0. \end{cases}$$

We get non-zero response at 2 places.

This high-pass filter is used for edge-detection which is used in a number of downstream tasks.

- Example: $h[n] = \dfrac{\delta[n] + \delta[n-1]}{2}$

This is a low-pass filter.

$$x[n] = \begin{cases} 1 & 0 \le n \le 3 \\ 0 & \text{else.} \end{cases}$$

$$y[n] = \begin{cases} 1/2 & n=0 \\ 1 & 1 \le n \le 3 \\ 1/2 & n=4 \end{cases}$$

In low-pass, locations of change are smoothed.

$y[n]$ is a "blurred" version of $x[n]$.

→ Why do we need LTI systems?

- They are excellent first order solutions to many problems.
- Applications:
  - Denoising
  - Edge detection.
  - Approximating visual systems.

Image filtering example:

$$h[i,j] = \frac{1}{9} \quad 0 \le i \le 2, \; 0 \le j \le 2 \qquad h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$I[i,j] = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Q. Can you compute $J(0,0)$?

$$J = I * h \qquad J[0,0] = \frac{1}{9}[1+2+5+6] = \frac{14}{9}$$

Q. Find $J[3,3]$.

$$\frac{1}{9}[11+12+15+16] = \frac{54}{9} = \underline{\underline{6}}$$

→ For edge detection, we use high pass filter. We want to detect the change in intensity.

18/03/2025
Tuesday

# Computer Vision

- Recap: Linear Image Filters
  - Optical Flow
  - Convolution Sum: $y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$

∵ System is LTI and any signal $x[n]$ can be written as $x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$

$$T\{x[n]\} = \sum_{k=-\infty}^{\infty} x[k]\, \underbrace{T\{\delta[n-k]\}}_{h[n-k]}$$

- $\delta[m,n] = \begin{cases} 1 & \text{if } m=0 \ \& \ n=0 \\ 0 & \text{o/w.} \end{cases}$

$$I[i,j] = \sum_u \sum_v I[u,v]\,\delta[i-u, j-v]$$

$J = I * h$

$$\boxed{J[i,j] = \sum_k \sum_m (I[k,m]).h[i-k, j-m]}$$

2D convolution sum.

## Low-pass filter: (Eg:- Mean, Gaussian)

- Smoothes the image.
- Reduces high frequency components, which means:
  - Reduces noise
  - Also blurs edges
- Think of it like averaging neighbourhood pixel values.

## High-pass filter: (Eg: Laplacian).

- Highlights high-frequency components like edges or sudden changes in intensity.

~~• Suppresses smooth~~

Q. What happens ~~yo~~ when you apply high-pass to noisy image?

A. Noisy image = high frequency everywhere.
Noise causes sudden small changes in pixel values across the image. That means lots of high-frequency content, even in places with no real edge.
Generally, when you apply high-pass filter, these sudden changes are amplified.
~~So, when you app~~ For noisy image, when high-pass filter is applied you end up detecting noise as false edges.
Real edge detection would be hampered.
So you get a black image.

→ Therefore, to denoise, we apply low-pass filter, because it removes unwanted high-frequency components. (Smooths out random noise).
Now you apply high pass filter. (like Laplacian).
This will help in responding to significant edges.

Laplacian Filter :- $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ → It is a second-order derivative ~~func~~ filter.

↓

Note that center gets subtracted by its neighbours.

→ When you reduce noise from 100 to 50 to 10, you can see small edges in the averaged image.

Optical Flow: ——→ used in motion estimation

Image acquisition:

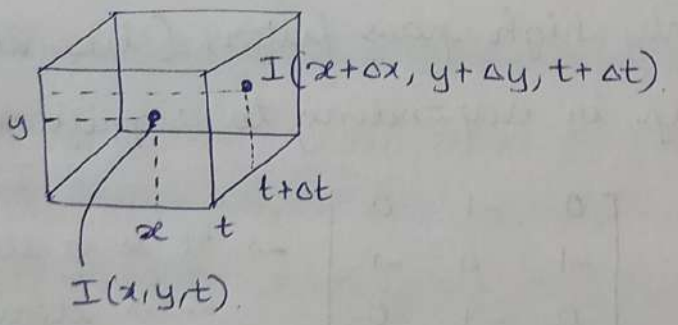
When a camera captures a photo:
Light (photons) hits the sensor. The light is converted to electric signals — intensities.
This creates a digital image.

→ $I(x, y, t)$ : Intensity of a pixel at time 't'.

→ Here we ~~are not~~ not working with static image but we are working with a video or image sequence over time.

→ Optical flow means "how pixels are moving between one frame and the next?".

→ We want to track how the intensity at point $(x, y, t)$ moves to a new location $(x+\Delta x, y+\Delta y)$ ~~$t+\Delta t$~~ at time $t+\Delta t$.




⇒ Let intensity $I(x, y, t)$ at spatial location $(x, y)$ and temporal location 't' move to spatial location $(x+\Delta x, y+\Delta y)$ at time $(t+\Delta t)$.

(Note : Intensity values its the same. Just changing in location.)

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,z) \quad \text{——— (1)}$$

Let's apply the Taylor series expansion to $I(x+\Delta x, y+\Delta y, t+\Delta t)$ at $(x,y,t)$

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,z) + \Delta x \cdot \frac{\partial I}{\partial x} + \Delta y \cdot \frac{\partial I}{\partial y} + \Delta t \cdot \frac{\partial I}{\partial t} + \text{higher order terms}$$

ignored for natural videos at reasonable rates (15 fmps).

(Because they are highly correlated, the changes move slowly and smoothly, so we can ignore higher order terms).

→ Using (1), we get, ——— (2)

$$I(x,y,t) + \Delta x \cdot \frac{\partial I}{\partial x} + \Delta y \cdot \frac{\partial I}{\partial y} + \Delta t \frac{\partial I}{\partial t} = I(x,y,z)$$

We are interested in how quickly the intensity is changing i.e. $\frac{\Delta x}{\Delta t}$ and $\frac{\Delta y}{\Delta t}$.

From (2),

$$\frac{\Delta x}{\Delta t} \cdot \frac{\partial I}{\partial x} + \frac{\Delta y}{\Delta t} \cdot \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0$$

$$\Rightarrow \boxed{I_x \cdot u + I_y \cdot v + I_t = 0}$$ ← Optical flow constraint

where $u = \lim_{\Delta t \to 0} \frac{\Delta x}{\Delta t}$, $v = \lim_{\Delta t \to 0} \frac{\Delta y}{\Delta t}$ } ← Velocity of $I(x,y,t)$

$$I_x = \frac{\partial I}{\partial x}; \quad I_y = \frac{\partial I}{\partial y}; \quad I_t = \frac{\partial I}{\partial t}.$$

→ We want to know u and v.

Since we have two variables, we need atleast one more equation other than this constraint to solve the problem.

21/03/2025
Friday

## Computer Vision

- Recap : Optical Flow (OF)
  - OFCE ( Optical flow constraint equation)
  - Smoothness constant.

### Optical Flow:

- $I(x, y, t) = I(x+\Delta x, y+\Delta y, t+\Delta t)$
- $I_x u + I_y v + I_t = 0$ → OFCE (Optical flow constraint equation)

$I_x = \frac{\partial I}{\partial x}$ $I_y = \frac{\partial I}{\partial y}$ $I_t = \frac{\partial I}{\partial t}$

$$u = \lim_{\Delta t \to 0} \frac{\Delta x}{\Delta t} \qquad v = \lim_{\Delta t \to 0} \frac{\Delta y}{\Delta t}$$

The optical flow vector is $\underline{f} = (u, v)$.

$u_x, u_y, v_x, v_y$ — flow gradients can be computed.

Velocities are not going to rapidly change spatially.

- Since OFCE is not sufficient to find $(u, v)$, we introduce the smoothness condition on the flow.

→ We use Horn-Schunck method.

We assume that neighbouring pixels tend to move in a similar way.

- This method minimizes the energy function.

$$E_o = \iint_{xy} (I_x u + I_y v + I_t)^2 \, dx \, dy.$$

In ideal situation, this is going to be zero. Because higher order terms are going to be zero in taylor expansion series. But this is not the case in real world.

$$E_s = \iint_{xy} (u_x^2 + u_y^2 + v_x^2 + v_y^2) \, dx \, dy.$$

$$u_x = \frac{\partial u}{\partial x} \quad u_y = \frac{\partial u}{\partial y} \quad v_x = \frac{\partial v}{\partial x} \quad v_y = \frac{\partial v}{\partial y}.$$

- Overall loss $E = E_s + \lambda E_o$.
  - $E_s$ → smoothness
  - $E_o$ → flow constraint

$E_o$ enforces brightness constancy.

$E_s$ encourages neighbouring motion vectors to be similar.

$\lambda$ controls balance between smoothness and flow constraint.

- How to solve?

We can set $\frac{\partial E}{\partial u} = 0$ ; $\frac{\partial E}{\partial v} = 0$ and solve for $u, v$.

→ In practice, we solve the discrete version of the problem.

$$E_0 = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (I_x u[i,j] + I_y v[i,j] + I_t)^2 \quad \text{——①}$$

$$E_s = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} ((u_x[i,j])^2 + (u_y[i,j])^2 + (v_x[i,j])^2 + (v_y[i,j])^2) \quad \text{——②}$$

$$\left.\begin{aligned} u_x &= \cancel{[u(i+1,j)}\ (u[i+1,j] - u[i,j]); \\ u_y &= (u[i,j+1] - u[i,j]); \\ v_x &= [v[i+1,j] - v[i,j]); \\ v_y &= [v[i,j+1] - v[i,j]) \end{aligned}\right\} \quad \text{——③}$$

Find $u, v$ s.t. $E = E_s + \lambda E_0$ is minimized. ——④

$$\frac{\partial E}{\partial u} = 0, \quad \frac{\partial E}{\partial v} = 0.$$

Substitute ③ in ④.

$$\frac{\partial E_0}{\partial u[i,j]} = 2[I_x u + I_y v + I_t] \cdot I_x$$

$$\frac{\partial E_0}{\partial v[i,j]} = 2[I_x u + I_y \cdot v + I_t] \cdot I_y$$

$$\frac{\partial E_s}{\partial u[i,j]} = \cancel{2[u-t}\ 2(u[i,j] - u_{avg}(i,j))$$

$$\frac{\partial E_s}{\partial v[i,j]} = 2(v[i,j] - v_{avg}(i,j))$$

24th march '25
Monday

# Computer Vision

Recap: $u = \lim_{t \to 0} \frac{\Delta x}{\Delta t}$ ; $v = \lim_{t \to 0} \frac{\Delta y}{\Delta t}$.

$I_x = \frac{\partial I}{\partial x}$ $I_y = \frac{\partial I}{\partial y}$ $I_t = \frac{\partial I}{\partial t}$ $\underline{f} = (u, v)$.

$I_x u + I_y v + I_t = 0$ ⟶ smoothness condition.

Ideally, the equation won't be zero, because we made some approximations in the Taylor series expansion.

$$E_o = \iint (I_x \cdot u + I_y \cdot v + I_t)^2 dx\, dy$$

OFCE ⟶ leads to aperture problem.

$$E_s = \iint (u_x^2 + u_y^2 + v_x^2 + v_y^2)\, dx\, dy.$$

Minimize $E = E_s + \lambda E_o$; i.e., ensure smoothness and accuracy.

$$E_s = \sum_{ij} \sum u_x^2 + u_y^2 + v_x^2 + v_y^2$$

$$u_x(i,j) = u(i+1,j) - u(i,j)$$

$$u_y(i,j) = u(i,j+1) - u(i,j)$$

$$v_x(i,j) = v(i+1,j) - v(i,j)$$

$$v_y(i,j) = v(i,j+1) - v(i,j).$$

Problem: Solve for $u(i,j)$ and $v(i,j)$.

Solution: Set $\frac{\partial E}{\partial u(i,j)} = 0$ ; $\frac{\partial E}{\partial v[i,j]} = 0$.

$$E = E_s + \lambda E_o$$

ensures smooth flow

Ensures that we satisfy our intensity constancy.

$$\bar{u}_{ij} = \frac{1}{4}[u[i,j+1] + u[i+1,j] + u[i-1,j] + u[i,j-1]]$$

$$\frac{\partial E}{\partial u} = 2(u - \bar{u}) + 2\lambda I_x (I_x u + I_y \cdot v + I_t)$$

$$\frac{\partial E}{\partial v} = 2(v - \bar{v}) + 2\lambda I_y (I_x \cdot u + I_y \cdot v + I_t)$$

Now, $\frac{\partial E}{\partial u} = 0, \quad \frac{\partial E}{\partial v} = 0.$

$$u(1 + \lambda I_x^2) + v(\lambda I_x I_y) + \lambda I_x I_t - \bar{u} = 0$$

$$v(1 + \lambda I_y^2) + u(\lambda I_x I_y) + \lambda I_y I_t - \bar{v} = 0$$

$$u(1+\lambda I_x^2)(1+\lambda I_y^2) + v(\lambda I_x I_y)(1+\lambda I_y^2) + (\lambda I_x I_t - \bar{u})(1+\lambda I_y^2) = 0$$

$$u(1+\lambda I_x^2)(\lambda I_x I_y) + v(\lambda^2 I_x^2 I_y^2) + (\lambda I_x I_t - \bar{u})(\lambda I_x I_y) = 0$$

$$u(1+\lambda I_x^2)(\lambda I_x I_y) + v(1+\lambda I_y^2)(1+\lambda I_x^2) + (\lambda I_y I_t - \bar{v})(1+\lambda I_x^2) = 0$$

$$v[1 + \lambda I_x^2 + \lambda I_y^2] + \lambda I_y I_t - \bar{v} + \lambda^2 I_x^2 I_y I_t - \lambda I_x^2 \bar{v} - \lambda^2 I_x^2 I_y I_t + \lambda I_x I_y \bar{u} = 0.$$

$$v = \bar{v} - \frac{\lambda(I_x \bar{u} + I_y \bar{v} + I_t) \cdot I_y}{(1 + \lambda I_x^2 + \lambda I_y^2)}$$

$$u = \bar{u} - \frac{\lambda(I_x \bar{u} + I_y \bar{v} + I_t) \cdot I_x}{(1 + \lambda I_x^2 + \lambda I_y^2)}$$

Q. Does this Horn-Schunk method converge?

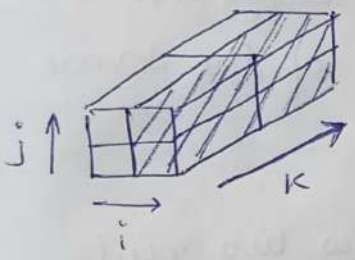
A. Yes, because it's based on minimizing a convex quadratic energy function.

Q. We don't know explicitly $I_x, I_y$ values. How do we get them?

A. Empirical Values.

OR by this method:

We can approximate $I_x$ using intensity values across two frames. (at time t and t+1).




The slices we consider are the shaded one and unshaded one.

$$I_x = \frac{1}{4}\left[I(i+1,j,k) + I(i+1,j+1,k) + I(i+1,j,k+1) + I(i+1,j+1,k+1)\right]$$

$$- \frac{1}{4}\left[I(i,j,k) + I(i,j+1,k) + I(i,j,k+1) + I(i,j+1,k+1)\right]$$

Recall: OFCE desired under the intensity constancy assumption.

25/03/2025
Tuesday

# Computer Vision

## Depth Estimation

→ Binocular stereo — it tells how machines perceive depth by mimicking human vision. We use two cameras to estimate how far away things are.

→ Two cameras seperated by a distance 'b'. (Distance b/w centers) — baseline 'b'.

→ Image planes are aligned coplanar.

→ Conjugate pair : Two points in different images that are the projections of the same point.

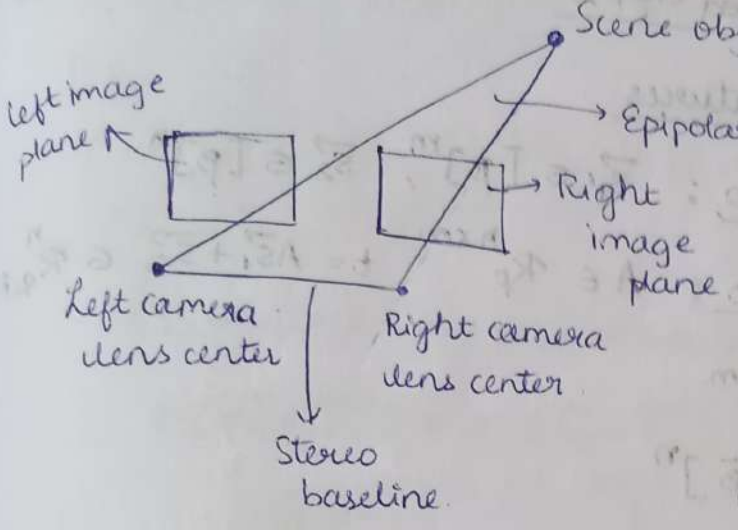
→ Disparity : Horizontal distance b/w two points in a conjugate pair, when the two images are superimposed

→ Epipolar plane : Plane formed by left camera center, right camera center and 3D point.

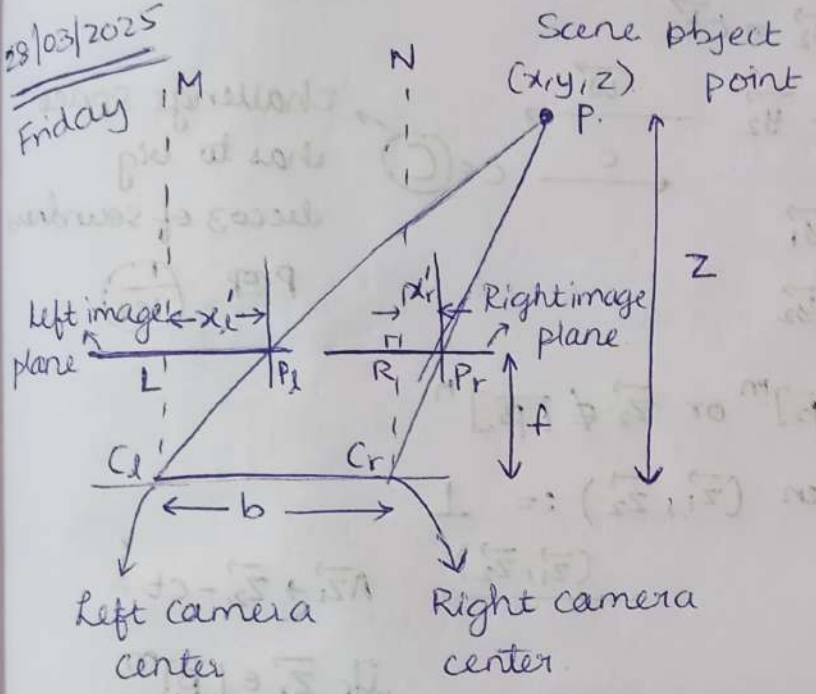
→ Epipolar line : Line where epipolar plane intersects with the image plane.

→ Feature in the left image at a particular y location will be present at the same y level in right image, just shifted horizontally, not up or down.

So, no vertical disparity.




28/03/2025
Friday




P is observed at points $P_l$ and $P_r$ in the left and right image planes.

$PMC_l$ and $P_l LC_l$ are similar.

$$\frac{x}{z} = \frac{x_l'}{f} \rightarrow P_l = \frac{P_l}{f}$$

$PNC_r$ and $P_r RC_r$ are similar.

$$\frac{x-b}{z} = \frac{x_r'}{f} \rightarrow P_r = \frac{P_r}{f}$$

$$\frac{x}{z} - \left(\frac{x-b}{z}\right) = \frac{P_l - P_r}{f} \Rightarrow \boxed{z = \frac{bf}{P_l - P_r}}$$

Depth is z. So we can see that depth can be recovered by knowing disparity of corresponding points.

## Relating ~~baseline~~ depth and disparity

$b \rightarrow$ baseline seperation

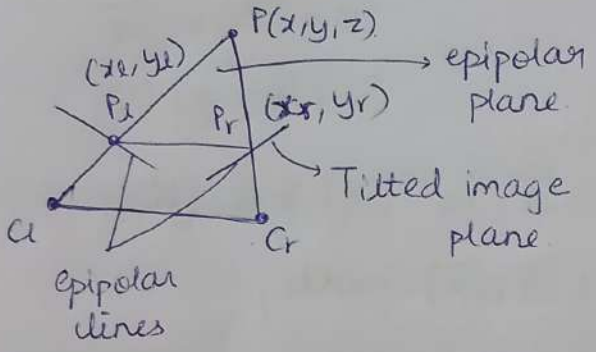
$f \rightarrow$ focal length of the camera.

$P_l - P_r \rightarrow$ disparity

→ We started with the assumption that this setup is ideal, i.e, the image planes are coplanar.

→ The image planes need not be coplanar in a non-ideal scenario.

epipolar plane: plane joining centers of the camera and the scene object.




Let $P_l = M_l \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

$P_r = M_r \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

Correspondence problem: For each point in the left image, find the corresponding point in the right image.

## Challenges in depth estimation

→ Image planes need not be coplanar.

→ One image plane could be rotated w.r.t the other.

→ Correspondence problem.

### ① Correspondence Problem

- find key points in left and right images.
  - classical methods like SIFT, SVCF, aSTFT etc
  - DL methods.

- Find corresponding key points using matching algos like RANSAC.
- ~~Inapo~~ Improve accuracy by our knowledge of 3D-geometry of the scene-specific epipolar geometry.
  - Require Intrinsic Camera Parameter.
- Scale Inversion Feature Transform (SIFT)

1/04/2025
Tuesday

## Computer Vision.

Recap: Challenges in depth estimation; correspondence problem.

- Scale Invariant Feature Transform (SIFT):
  - Scale space construction — DoG → Difference of Gaussians
  - Key points from extrema.
  - Refining key points
  - Orientation assignment.
  - Keypoint description.

Depth and disparity have inverse relation.

→ ~~Eo~~ SIFT algo is used in stereo vision or matching problems where we want to find corresponding points — called matching pairs,
{basically, left and right camera view of of the same scene.}

How to find keypoints in both images?

Goal is:

To detect key points in an image that are:

- Distinctive — easy to match across images
- Invariant to scale: Means whether image is zoomed in or out, same keypoint can be detected.
- Invariant to rotation: Even if image is rotated, same keypoint ....

- First, we define keyspace

  Keypoints are defined in 3D space: $(x, y, \sigma)$

  $x, y$ = coord  $\sigma$ = scale at which keypoint is detected

- Create scale space:

  Take image, apply low-pass filter. Blurred images are created. Each set of these blurred images is called octave.

  → In octave, image is downsampled by a factor of 2 in both width and height

- (DoG): Difference of Gaussians : Bandpass Filtering:

  Bandpass filters are obtained by subtracting two Gaussian blurred images.

$$DoG = G(x, y, \sigma_2) - G(x, y, \sigma_1)$$

** Band pass regions contain keypoints.

- Now we check for local extrema points in band pass regions.

How do check for extrema?

We compare each pixel to its 26 neighbours.

26 neighbours?

⇒ ① 8 in the current scale
② 9 in the scale above
③ 9 in the scale below.

The extremas we get are potential keypoints.

→ Now we got keypoints extremas, but some might not be useful. Like they might be lying on edges. So we need to do refining.

→ We do pruning. This will prune spurious points. It discards keypoints with low contrast and also those which lie on edges.

→ We make keypoints rotation invariant and scale invariant.

→ How to make keypoints rotation invariant?

⇒ * Around each keypoint, we calculate gradient orientation in neighbourhood.
* We construct histogram of orientations.
* The peak of histogram is assigned as the dominant orientation.
(Multiple orientations are also possible)

→ Till now, we have for each keypoint

(x, y, scale, orientation)

Q) Why are we making keypoints invariant to scale and orientation?

Ans: ⇒ Small details get lost when we shrink an image. We want to detect keypoints at multiple scales and ensure that they are not lost during downsampling or zooming.

→ Also, when an image is rotated, the orientation of keypoints changes.
Our algo already knows each keypoint's orientation, so it subtracts this angle.
So makes it rotation invariant.

→ Now we look at Histogram of Gradients (HoG).

Take a keypoint, around it take a 16×16 pixel patch. ~~(adjoin with~~

Divide this into 4×4 sub-regions → 16

Around each subregion, we compute a 8-bin histogram (like for 8 directions).

So $16 \times 8 = 128$.

Feature Vector is 128-dimensional. (normalized)

This is illumination invariant.
(Intensity scaling won't effect).

→ ** Keypoint now has:

- (x, y) → location.
- Scale
- Orientation.
- HoG Descriptor → (128-length).

→ Now do matching, take keypoints from two images and compare feature vectors.

Eg: Suppose we want to detect a clock.
We have a database of objects with precomputed SIFT features.
Now for our image (query), compute SIFT features.
Match these with that of database images.
If good no. of matches, object is present.

Summary:

→ Keypoint candidates are the maxima of DoG-filtered responses.
DoG response to image is difference of Gaussians.

→ Why do we need band-pass filtered regions?
Because keypoints are found there.

→ A DoG-filtered sample is an extrema if largest or smallest of 26 surrounding pixels, in scale-space.

→ The gradient magnitudes and orientations of patch around the keypoint is computed.

→ Then we normalize the histogram.
↓
Why? To be invariant to illumination changes.

Matching:

Now we have a set of keypoints and their representations. We compare the values.

Goal: We want keypoints that are invariant to resolution.

April 4th 2025
Friday

# Computer Vision.

- Recap: SIFT algorithm.
- Camera matrix — Notation: Inhomogenous, Homogenous, Augmented.

— Extrinsic — R, t : Rotation and Translation.
— Intrinsic — K ; camera properties — focal length, scaling, center location.
— Projection — M — K[R, t].

→ After detecting keypoint, we now want representation. We use HoG.

Take a patch (neighbourhood). At each pixel compute gradient magnitudes and orientation. Quantize orientation into 8 bins. 45° range ($\frac{360}{45} = 8$)

Finally — feature vector (HoG).

## Camera Parameters

Goal: We want to project a point from 3D space (real world) into 2D pixel space of image. (How object is seen on camera?)

Points to note:

→ It accomodates rotation as well as translation in our camera matrix.

→ Extrinsic Parameters: Defined outside of the camera. Rotation, Translation.

→ Intrinsic Parameters: Built into the camera. Focal length, Resolution of the sensor, Center of

image. $\left(\frac{\text{img width}}{2}, \frac{\text{height}}{2}\right)$ Eg:- 640x480 image = (320,240).

→ Ransac ~~Problem~~ algorithm helps in like when we have keypoints (feature vectors HoG), we need to match across images. RANSA filters out bad matches.

→ How does epipolar line help?

Given a key point in the left image, the corresponding point in the right image must lie on the epipolar line.

So instead of searching entire image; we only search along epipolar line (complexity is reduced.)

- 3D to 2D projection.
  - Expression
  - Example:

intrinsic → $K = \begin{bmatrix} 800 & 0 & 320 \\ 0 & 800 & 240 \\ 0 & 0 & 1 \end{bmatrix}$ extrinsic → $R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $t = \begin{bmatrix} 0 \\ 0 \\ -5 \end{bmatrix}$

Where does the point $X = [2, 3, 10, 1]^T$ get projected on the image?

- Let's operate in $\mathbb{R}^3$.

$\underline{P} := [x \; y \; z]^T$ — Inhomogenous Representation.

$\tilde{\underline{P}} \in P^3 = \mathbb{R}^4 - (0,0,0,0)$; $[x, y, z, w]^T$

— Homogenous Representation.

Here $w$ is scaling factor.

$\overline{\underline{P}} \in \mathbb{R}^4 = [x, y, z, 1]^T$; Augmented Representation.

- K: Intrinsic Parameters:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

→ skew: non-zero for non-rectangular pixels (distorted)

→ location of the center of the image. (in pixels)

→ focal length of the camera along the x and y dimensions.

- Extrinsic parameter:

$[R\ t]$ → Translation matrix (3×1)

↓ Rotation matrix (3×3)

- Overall projection matrix

$$M = K[R\ t]$$

K: Intrinsic camera parameter

[R t]: Extrinsic.

[R t] → 3 rows, 4 columns.

$$\bar{i} = M \cdot \bar{P}$$

$$M\bar{x} = \begin{bmatrix} 800 & 0 & 320 \\ 0 & 800 & 240 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -5 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 3200 \\ 3600 \\ 5 \end{bmatrix}$$

$\bar{P}$

Scaling factor

$$\bar{P} = \begin{bmatrix} 640 \\ 720 \\ 1 \end{bmatrix}$$

April 7th 2025
Monday

# Computer Vision

Recap: $M = K[R\ t]$ project a 3D location to a 2D location.

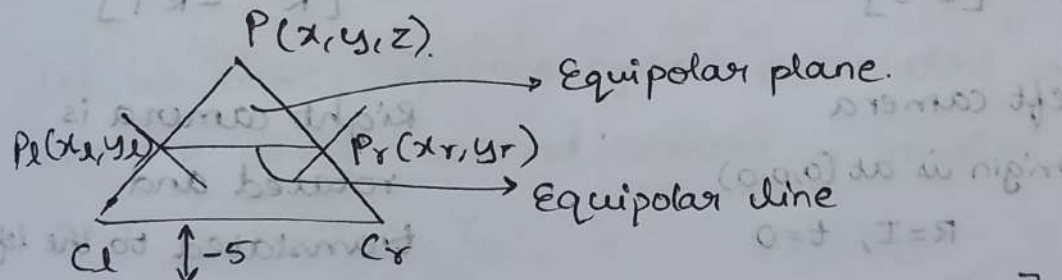
- How is K defined? $K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$

Today:

- Pinhole camera model: $x = \frac{x f}{z} + c_x$
- Putting it together:
  - Essential matrix.
  - Epipolar line.
  - Corresponding points.

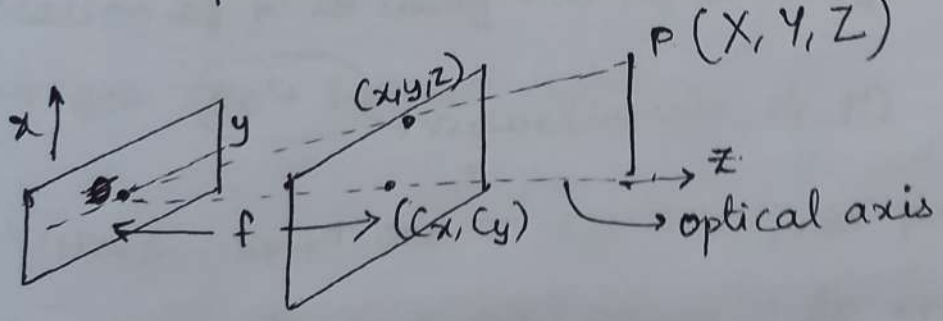
Depth and disparity are inversely related.

Suppose there are pair of images. We can use DL models (monocular) to circumvent the region where the second can be found.




$$P_l = M_l \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad P_r = M_r \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- Assume a pinhole camera model.




$$\frac{x'}{f} = \frac{X}{Z} \Rightarrow x' = \frac{Xf}{Z}$$

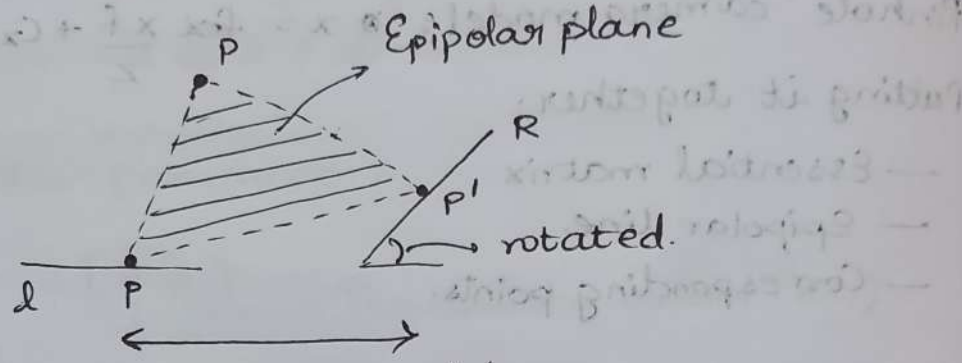
Relative to the image origin,

- $x' = x - C_x$.

$$x - C_x = \frac{Xf}{Z}$$

$$\Rightarrow x = \frac{Xf}{Z} + C_x.$$

- Tying it together




$M = K[R \; t]$ → $R = I, t = 0$

$M = K[I \; 0]$ → $K = I$

$M = [I \; 0]$

$M' = K'[R^T \; -R^T T]$

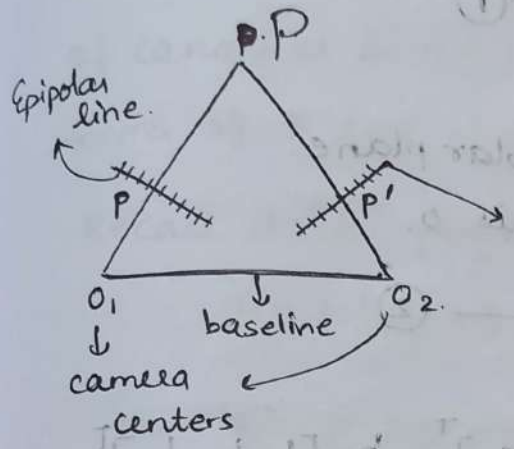
Left camera.
Origin is at (0,0,0)
$R = I, \; t = 0$

Right camera is rotated and translated to the left
$T \rightarrow$ baseline vector.
Translated by.

8th Apr 2025
Tuesday.

# Computer Vision

Recap: Convex, projection matrix.
- Essential matrix.
- Fundamental matrix.




Two cameras p and p' are observing the same 3D point P.

- We don't know the exact location of P.

Using $O_1, O_2$ and image point p, we can define epipolar plane. Using epipolar planes, we get the epipolar lines.

$$M = K[R, T].$$

let us take reference system associated to left camera. and second camera is rotated by R and translation T.

So, we get camera projection matrices as

$$M = K[I \;\; 0] \qquad M' = K'[R^T \;\; -R^T T]$$

→ location of p' using this reference system, we get $(Rp' + T)$ → vector (location of p').

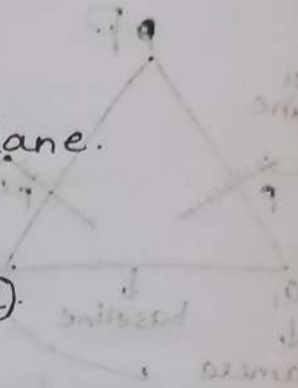
→ Vectors $Rp'+T$ and $T$ lie in epipolar plane.

So $(T \times (Rp'+T))$ → normal to epipolar plane.

We assume the cameras are canonical — focal lengths are 1.

$$T \times (Rp' + T) = T \times Rp' \quad \text{[normal to the epipolar plane]} \quad \text{— (1)}$$

($T$: extrinsic params of right camera)



→ We know $p$ lies on the epipolar plane.
So dot product with normal is 0.

$$p^T \cdot (T \times Rp') = 0 \quad \text{— (2)}$$

Generally, if $a = [a_1\ a_2\ a_3]^T$, $b = [b_1\ b_2\ b_3]^T$

$c = a \times b = [a_x]b$ can be written as

$$c = a \times b = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

→ Standard property of cross product.

$T \times Rp'$ can be written as ~~$[T_x][Rp']$~~ $[T_x]Rp'$ where

$$T_x = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

Eqn (2) can be written as

$$p^T \cdot [T_x] R p' = 0 \quad \text{— (3)}$$

$$\boxed{p^T \cdot E \cdot p' = 0}, \quad \text{where } E = [T_x]R$$

~~Ess~~ → Essential matrix

**Essential Matrix $E = [T_x]R$**

→ Eqn (3) constraint is used to search for corresponding points.

- The above analysis assumed canonical cameras i.e. $K = K' = I$.

more general case is not-canonical.

Here also, we will have $p$ and $p'$ (irrespective of canonical or not). Now, to get the canonical forms of $p$ and $p'$ when we have $K$ and $K'$:

Recall that projection matrices:

$$M = K[I \;\; 0] \qquad M' = K'[R^T \;\; -R^T T]$$

So, ~~p~~ $p$ becomes $K^{-1}p$., $p'$ becomes $K'^{-1}p'$.

→ If we don't have canonical cameras, we find the effective canonical representations of $p$ and $p'$ as follows:

$$p_c = K^{-1}.p \quad \text{——} (4)$$

$$p'_c = K'^{-1}p' \quad \text{——} (5)$$

→ Let's plug (4) and (5) into (3).

$$p_c^T . E . p'_c = 0.$$

$$p^T (K^{-1})^T E . (K'^{-1}p') = 0$$

$$\boxed{p^T F p' = 0} \quad \text{——} (6)$$

where $F = (K^{-1})^T E K'^{-1}$

** $\boxed{\text{Fundamental matrix } F = (K^{-1})^T E K'^{-1}}$

→ F can be estimated directly from "known" points.

→ If we know F, then simply knowing a point in an image gives us a constraint (epipolar line) of the corresponding point in other image.

→ Difference between this setup and the original setup we discussed few ~~tech~~ classes before is that:
Before we considered image planes were coplanar with no rotation.

YOLO → You only look once.
↳ Globally used for object detection.

11th Apr 2025
Friday

## Computer Vision

### Object Detection

IoU Score : (Intersection Over Union)
It basically tells how much overlap does our box have with the desired object.

Q. How can you use the classifier to detect an object?

Take image patches and pass it through the classifier. We have patch level labels. We can even pass overlapping patches. We pick all the patches where the overlapping probability is high.

# YOLO

→ A popular deep learning based object detection algorithm.

→ Tries to identify the potential regions of finding the object.

→ Older models like RCNN, Fast RCNN, Faster RCNN lacked in speed although errors were less.

→ YOLO formulates object detection as a regression problem. It regresses:

- (x, y) → object center location
- width (w) and height (h) of bounding box
- Confidence Score (class probabilities) of the ~~Class Probabilities~~ bounding box around the object (class).

→ ~~Each bounding box outputs a~~

→ Divide the input image into SxS grid.

Each grid cell is further divided into B blocks.

Each block is assigned a class conditional probability.

So, each grid cell predicts:

- B bounding boxes
  - Each box: x, y, w, h, confidence — 5 parameters
- C class probabilities (C → no. of classes)

So output tensor has shape:

$$S \times S \times (B \times 5 + C)$$

→ Bounding box outputs confidence score as:

** Confidence = $Pr(\text{Object}) \times IOU^{\text{truth}}_{\text{pred}}$

$Pr(\text{Object})$: prob. that object is present.

$IOU^{\text{truth}}_{\text{pred}}$: Intersection over Union b/w predicted box and ground truth.

→ Issue : Resolution limitation.
The model ~~has~~ is limited, like no. of objects it can be ~~to~~ able to detect and where.
If two objects have centers in the same grid cell, one might get missed.

→ We use rectang~~ules~~ for bounding boxes because they have few parameters (x, y, w, h).

→ Combined regression + classification problem for each box.

→ Q. What can be a candidate model to solve the problem?
Classifier - architecture model (like CNN) classifier.

→ Unified detection of all objects in an image.
↓
Does not work on individual proposals.

→ Performs global reasoning about the image for prediction.
(like all objects simultaneously across the entire image).

→ If an object center falls into a grid cell, that grid cell is assigned the responsibility for detecting that object.

→ Each grid cell predicts B bounding boxes and the confidence for the bounding boxes.

→ Bounding box confidence depends on the IOU between ground truth prediction and the probability of an object being present.

→ During training, bounding boxes with the highest confidence are retained.

→ All other boxes are suppressed.

→ During inference, multiple bounding boxes may be predicted.

We use non-maximal suppression to eliminate overlapping boxes with lower confidence.

It basically keeps only the box with highest confidence for overlapping predictions.

→ NMS is used to resolve multiple detections.

## Non-maximal suppression

It removes duplicate predictions for the same object.

First, it sorts boxes by confidence. It picks the box with the highest confidence, and suppresses all boxes with high IOU overlap.

## Cost function:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] +$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] +$$

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{noobj} (C_i - \hat{C}_i)^2$$

$$+ \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (P_i(c) - \hat{P}_i(c))^2$$

This is categorical

this term tells what type of object it is.

This term tells us the confidence, or what area of the box is covered with object.

(Intersection of particular box with ground truth.)

↳ This is geometrical.

$\lambda_{noobj}$ = weighting factor, set to a small value so that no-object boxes don't dominate.

15th April 2025
Tuesday

# Computer Vision

$S \times S = (7 \times 7)$

$B = 2$, → PASCAL VOC dataset

$C = 20$ (no. of object classes)

$(x, y)$, $(w, h)$ → normalized w.r.t dimensions of image.

$C_i$ of $B_i$ = $Pr(\text{object}) \times IOU^{\text{truth}}_{\text{pred}}$

$P(C) = P(\text{class} = c \mid \text{object})$ — class conditional probability.

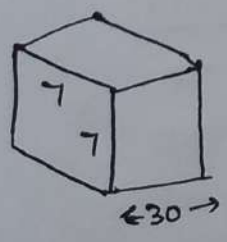
$1^{obj}_{i,j}$ = $j^{th}$ bounding box predictor for the $i^{th}$ grid cell responsible for the object detection.

→ In the loss function, we take square-root over $w_i$ and $h_i$ because we want to suppress the dominance of width and height.

→ Bounding box need not lie within the cell. There is no such restriction.

→ Why multiple boxes?
more the no. of candidates for prediction; the better would be the overlap with the object.

→ If one box is large and another is smaller but fits the object tighter, the smaller one will have a higher IOU and will be favoured.




We know output tensor has shape $S \times S \times (B \times 5 + C)$.

$7 \times 7 \times (2 \times 5 + 20)$.

$\Rightarrow 7 \times 7 \times 30$.

21st April '25
Monday

# Computer Vision

## Image Segmentation

Q. How is image segmentation different from object detection?

A. In image segmentation, each pixel in an image is assigned a label.
In object detection, instead of assigning probability to every image, we try to assign probability to every bounding box in object detection.

- **Semantic segmentation:**
  Partitions image into semantically meaningful partitions. Assigns each pixel a class label. Does not distinguish blw different instances of the same object.

- **Instance segmentation:**
  Assigns IDs to each pixel. Distinguishes objects of the same class.

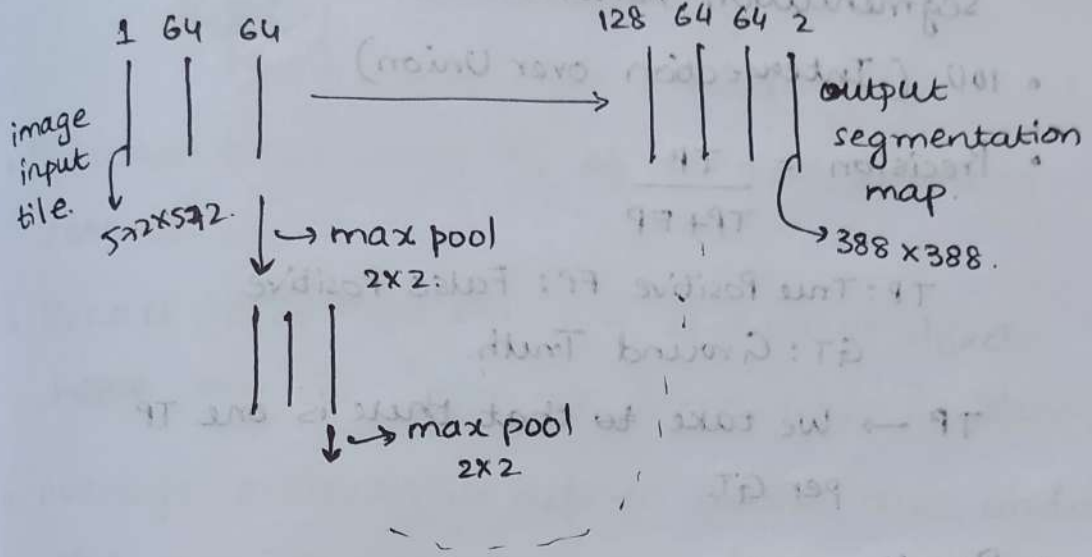
- **Panoptic Segmentation:**
  Combines semantic and instance segmentation. Assigns both class and ID.

U-Net is used for semantic segmentation
↳ Image segmentation.

22nd April '25
Tuesday

# Computer Vision

→ UNet architecture is used for semantic segmentation.




→ Input is image of size 572 x 572.
Output is a segmentation map of size 388x388.

→ left side of this UNet Arch: Encoder (Contracting Path).
This basically learns what is present.

→ Right side: Decoder (Expansive path).
It learns the location.

You start with 1 input channel. Finally you have 2 output channels. (binary classification).
↓
background or foreground).

28th Apr 2025
Monday

# Computer Vision.

- UNet Model
- ~~Detecting~~ Measuring Detection and Segmentation Performance.
- IoU (Intersection over Union)
- Precision = $\frac{TP}{TP+FP}$

  TP: True Positive FP: False Positive

  GT: Ground Truth.

  TP → We take ~~to~~ that there is one TP per GT.

- Recall = $\frac{TP}{\text{Total no. of objects.}}$
- Average Precision (AP): Area under the precision-recall curve.
- mAP (Mean Average Precision): ~~Average~~ Average AP values over multiple classes.

Example: 'Dog' class,

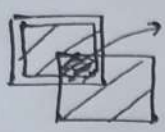
4 GroundTruth objects, 5 predictions with 'Dog' label.

- IoU tells about the overlapping b/w the predicted bounding box and ground truth.
- Suppose we take (IOU ≥ 0.5) to say it as TP.

| Pred | Confidence | IoU | TP/FP |
|---|---|---|---|
| 1 | 0.95 | 0.9 | TP |
| 2 | 0.9 | 0.3 | FP |
| 3 | 0.8 | 0.65 | TP |
| 4 | 0.7 | 0.7 | TP |
| 5 | 0.6 | 0.25 | FP |

→ we should ~~have~~ do overlap tiling for smooth segmentation.

. IoU.




$IoU = \frac{\text{blue}}{\text{blue}}$

. Precision tells what % of our detections are correct.

. Recall tells how much % of actual objects ~~have~~ ~~we~~ ~~to~~ have we been able to identify.

→ Average precision is defined as the area under the precision-recall curve.

- Mean AP $= \frac{1}{Nc} \sum_{d=1}^{Nc} AP_d$

- Confidence : Actual probability assigned to obj.class $\times$ IOU.

- ~~TP~~ ~~FP~~ TP/FP (IOU $\geq$ 0.5)

- These are computed w.r.t the closest GT available.

Eg:- 4 GT objects, 5 Preds with 'Dog' label.

| Pred | Conf | IOU | TP/FP | ~~RP~~ Prec | REC |
|---|---|---|---|---|---|
| 1 | 0.95 | 0.9 | TP | $\frac{1}{1+0}=1$ | 1/4 |
| 2 | 0.9 | 0.3 | FP | $\frac{1}{1+1}=0.5$ | 1/4 |
| 3 | 0.8 | 0.65 | TP | $\frac{2}{2+1}=\frac{2}{3}$ | 2/4 |
| 4 | 0.7 | 0.7 | TP | $\frac{3}{3+1}=\frac{3}{4}$ | 3/4 |
| 5 | 0.6 | 0.25 | FP | $\frac{3}{3+2}=\frac{3}{5}$ | 3/4 |

* 'Recall' can never ~~have~~ be greater than 1.
We can never flag all the objects as TP.
(Each GT cannot have more than 1 TP)

| Recall | Precision |
|---|---|
| 0.25 | 1 |
| 0.25 | 0.5 |
| 0.5 | 0.67 |
| 0.75 | 0.75 |
| 0.75 | 0.6 |

$Prec(r=0.25) = \max(1, 0.5, 0.6, 0.67, 0.75)$

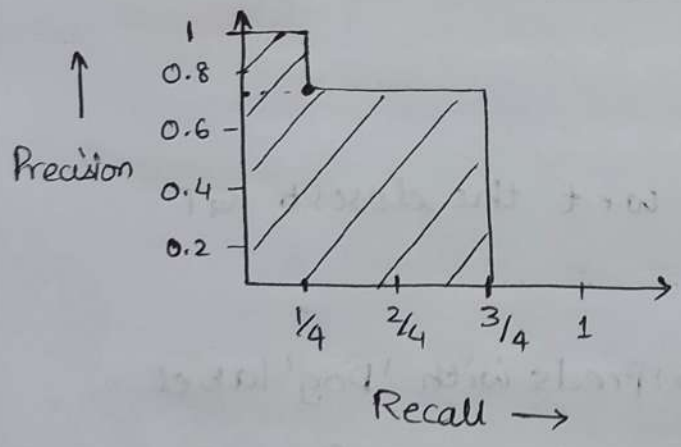
$= 1$

$Prec(r=0.5) = \max(0.67, 0.75, 0.6)$

$= 0.75$

$Prec(r=0.75) = \max(0.75, 0.6)$

$= 0.75.$




Avg precision = Area under the precision-recall curve.

$AP = (0.25-0) \times 1 + (0.75-0.25) \times 0.75$

$= 0.25 + 0.5 \times 0.75$

$= 0.625$

Q. What happens if we consider ~~P(r=1)~~ Prec(r=1)?
It does not have a value in the table.

$P(r=1) = 0$ //

$AP_{dog}^{0.5} = 0.625$

Q. Using this metric, when does the model perform ideally?

A. When it detects all true objects (high recall) and when it makes few FP (high precision).

Area under curve is close to 1.

In ideal case, model has AP of 1 for each class.

→ Average precision (AP) takes care of both precision and recall in one-shot.

→ If model is really good, then the fall of mAP values shouldn't be very high.

It should have consistently high AP across all classes. And also, even if you change the IoU threshold, good model should maintain decent AP.

29th Apr '25
Tuesday

## Computer Vision

→ A high no. of false positives lowers precision.

→ A detector with many FP is considered poor in terms of precision.

→ Non maximum suppression is applied after detection (YOLO uses this).

Goal: Keep one bounding box per actual object.

## NMS

→ Assign confidences to all bounding boxes.

→ For each class, sort all detections by confidence score.

→ Pick the highest confidence box – best match.

→ Now for every other box:

- Compute IoU with the best box.
- If IoU > threshold, discard that box (becoz they are overlapping too much).

→ So basically, it reduces duplicate detections for the same object.

→ Confidence is used for filtering predictions, ranking detections and plotting precision-recall curves.

→ To evaluate whether a prediction is TP, we first match predicted box to a ground truth box using IoU.

If IoU ≥ threshold its TP. Else it is a FP.

Also, we only want one predicted box per ground truth box. So multiple matches count as extra FPs.

→ Can these terms be computed at inference time?

Pr(class) → Yes.

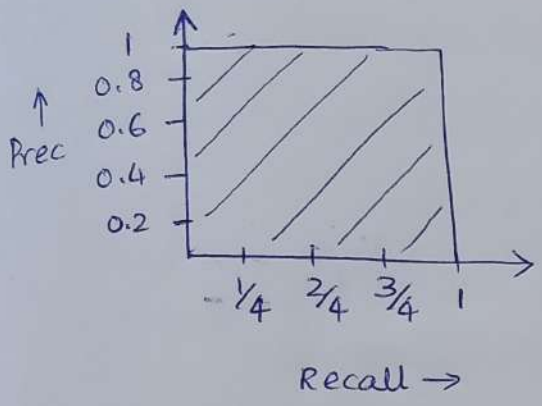
NMS → Yes.

But FP, TP need ground truth, so they can't.

Example: IoU threshold = 0.5, 4 GT, 5 preds.

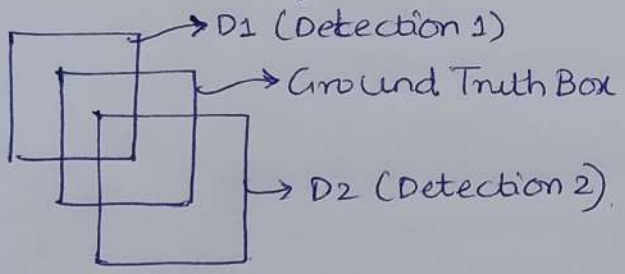
| Confidence | IoU | TP/FP | Prec | Recall |
|---|---|---|---|---|
| 0.9 | 0.95 | TP | 1 | 1/4 |
| 0.8 | 0.85 | TP | 1 | 2/4 |
| 0.6 | 0.7 | TP | 1 | 3/4 |
| 0.5 | 0.6 | TP | 1 | 1 |
| 0.4 | 0.45 | FP | 4/5 | 1 |

Prec-Recall curve:




AP = 1.

→ Now comparing, two detections with Ground truth.




P(dog) = 0.9, IoU = 0.8 } D1

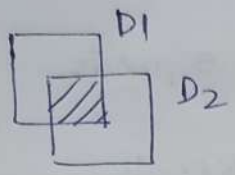
P(dog) = 0.85, IoU = 0.81 } D2

Which one do we consider as the best confident box?

Conf of D1 : 0.9 × 0.8 = 0.72 Conf of D2 = 0.85 × 0.81 < 0.72.

So, we designate D1 as our GT.

Now we check the intersection of D1 and D2.




When is D2 considered as a potential candidate?

If D1 and D2 are overlapping too much, i.e IoU of D1 and D2 is beyond a threshold, say 0.5, we discard it, since D1 and D2 in this case would potentially be duplicates.

If IoU < 0.5, then even D2 is considered as a GT.