ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

MÔN HỌC: THÔNG TIN DI ĐỘNG (EE5019)

BÁO CÁO BÀI TẬP LỚN

ƯỚC LƯỢNG KÊNH TRUYỀN VÔ TUYẾN

Giảng viên hướng dẫn: PGS. TS. Hà Hoàng Kha

Sinh viên thực hiện: Phạm Khánh MSSV: 2011391

Lê Nguyên Khôi MSSV:

Nhóm P01 – Nhóm 2

Tp. Hồ Chí Minh – 2024

Mục lục

1 Giới thiệu tổng quan

Trong các hệ thống thông tin vô tuyến, kênh truyền đóng vai trò quan trọng do tín hiệu truyền qua kênh thường bị méo bởi các hiệu ứng như nhiễu, fading (suy hao), và tán xạ. Để khôi phục được tín hiệu đã truyền, đầu thu cần phải ước lượng kênh (channel estimation), tức là xác định đặc tính của kênh truyền để bù đắp sự méo dạng của tín hiệu. Việc ước lượng kênh hiệu quả giúp cải thiện chất lượng tín hiệu thu được và tăng cường hiệu suất của hệ thống truyền thông.

Ước lượng kênh thường được thực hiện bằng cách sử dụng các ký tự pilot - các tín hiệu đã biết trước, được chèn vào trong khung thời gian hoặc tần số. Dựa trên các giá trị pilot nhận được, hệ thống sẽ tính toán để ước lượng đặc tính của kênh tại các vị trí khác.

Các phương pháp ước lượng kênh phổ biến có thể kể đến là:

1. **Least Squares (LS)**: Phương pháp đơn giản, chỉ sử dụng các ký tự pilot mà không cần thông tin thống kê của kênh. Tuy nhiên, độ chính xác không cao trong các môi trường có nhiễu.
2. **Minimum Mean Square Error (MMSE)**: Sử dụng thông tin thống kê của kênh để cải thiện độ chính xác so với LS, nhưng có độ phức tạp tính toán cao và yêu cầu thông tin đầy đủ về kênh.
3. **Phương pháp dựa trên học sâu (Deep Learning)**: Gần đây, các phương pháp ước lượng kênh dựa trên học sâu được phát triển, tận dụng các mạng nơ-ron để tự động học và ước lượng kênh từ dữ liệu. Phương pháp này hứa hẹn hiệu suất cao hơn trong các môi trường phức tạp và nhiễu mạnh, nhưng yêu cầu quá trình huấn luyện mô hình phức tạp.

Ước lượng kênh truyền vô tuyến đóng vai trò quan trọng trong nhiều ứng dụng thực tế, đặc biệt là trong các hệ thống truyền thông không dây. Một trong những ứng dụng phổ biến nhất là trong mạng di động như LTE (4G) và 5G NR. Trong các hệ thống này, ước lượng kênh được sử dụng để xử lý tín hiệu tại các trạm gốc và thiết bị đầu cuối, giúp đảm bảo truyền dữ liệu với độ chính xác cao, từ đó cải thiện chất lượng cuộc gọi, tốc độ truyền tải dữ liệu, và giảm thiểu hiện tượng mất gói tin.

Bên cạnh đó, ước lượng kênh cũng rất quan trọng trong hệ thống Wi-Fi, đặc biệt với các chuẩn Wi-Fi tiên tiến như Wi-Fi 6 và Wi-Fi 7. Nó giúp điều chỉnh quá trình truyền tải dữ liệu qua các băng tần cao, đảm bảo kết nối nhanh và ổn định hơn, đồng thời giảm nhiễu và tăng cường hiệu suất mạng. Điều này đặc biệt hữu ích trong các môi trường nhiều thiết bị kết nối như văn phòng hoặc nhà thông minh.

Trong các hệ thống liên lạc vệ tinh, ước lượng kênh giúp tối ưu hóa tín hiệu khi truyền qua khoảng cách xa và qua các môi trường khắc nghiệt như không gian và bầu khí quyển.

Nhờ đó, tín hiệu truyền từ vệ tinh đến các trạm mặt đất hoặc ngược lại có thể được phục hồi chính xác, đảm bảo truyền tải dữ liệu an toàn và tin cậy, đặc biệt là trong các ứng dụng điều khiển từ xa hay viễn thám.

Trong lĩnh vực giao tiếp giữa các phương tiện, ước lượng kênh hỗ trợ mạnh mẽ cho hệ thống V2X (Vehicle-to-Everything), bao gồm giao tiếp giữa các phương tiện với nhau (V2V) hoặc giữa phương tiện và hạ tầng (V2I). Với tốc độ cao và điều kiện môi trường thay đổi liên tục, việc ước lượng kênh chính xác giúp đảm bảo truyền thông ổn định và kịp thời, tăng cường an toàn giao thông và hỗ trợ các hệ thống xe tự lái.

Một ứng dụng quan trọng khác là trong các hệ thống MIMO (Multiple Input Multiple Output), đặc biệt với 5G. Trong các hệ thống này, việc sử dụng nhiều anten để truyền tải và nhận tín hiệu yêu cầu một mô hình ước lượng kênh hiệu quả để tối ưu hóa băng thông và tốc độ truyền dữ liệu. Ước lượng kênh giúp hệ thống khai thác hiệu quả sự đa dạng của các kênh vô tuyến mà không tăng thêm yêu cầu về băng thông.

Ngoài ra, ước lượng kênh còn có ứng dụng trong các hệ thống radar và truyền thông quân sự, nơi tín hiệu phải đi qua những điều kiện môi trường phức tạp và bị nhiễu. Ước lượng kênh giúp tăng độ chính xác trong việc tái tạo tín hiệu, từ đó cung cấp thông tin liên lạc đáng tin cậy và an toàn trong các nhiệm vụ quân sự hoặc do thám.

Cuối cùng, trong các hệ thống Internet vạn vật (IoT), ước lượng kênh giúp các thiết bị cảm biến không dây hoạt động hiệu quả hơn trong các môi trường có nhiều nhiễu và thay đổi nhanh chóng về điều kiện kênh. Điều này đặc biệt quan trọng với các ứng dụng truyền thông tầm xa như LoRa, giúp duy trì kết nối ổn định và tiết kiệm năng lượng trong các mạng lưới cảm biến diện rộng.

Như vậy, ước lượng kênh truyền vô tuyến không chỉ là một phần không thể thiếu của các hệ thống truyền thông hiện đại mà còn là yếu tố quyết định hiệu suất và độ tin cậy trong nhiều ứng dụng khác nhau, từ mạng di động, Wi-Fi, đến các lĩnh vực chuyên biệt như liên lạc vệ tinh và hệ thống IoT.

2 Mô hình hệ thống

Mô hình hệ thống ước lượng kênh truyền vô tuyến mô tả quá trình một thiết bị thu nhận ước lượng chính xác về đặc tính của kênh vô tuyến giữa máy phát và máy thu. Quá trình này giúp thiết bị bù đắp được các ảnh hưởng của môi trường truyền thông (như nhiễu, suy hao, tán xạ) để tái tạo lại tín hiệu gốc.

2.1 Mô hình kênh vô tuyến

Kênh truyền vô tuyến có thể được mô tả bằng phương trình đầu vào – đầu ra như sau:

$$\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N} \tag{1}$$

Trong đó:

- $\boldsymbol{Y}$: Tín hiệu nhận được tại thiết bị thu.
- $\boldsymbol{X}$: Tín hiệu đã truyền đi từ máy phát.
- $\boldsymbol{H}$: Ma trận kênh vô tuyến biểu diễn đặc tính của kênh (suy hao, trễ, tán xạ).
- $\boldsymbol{N}$: Nhiễu trắng Gaussian cộng (AWGN), mô tả nhiễu ảnh hưởng đến tín hiệu.

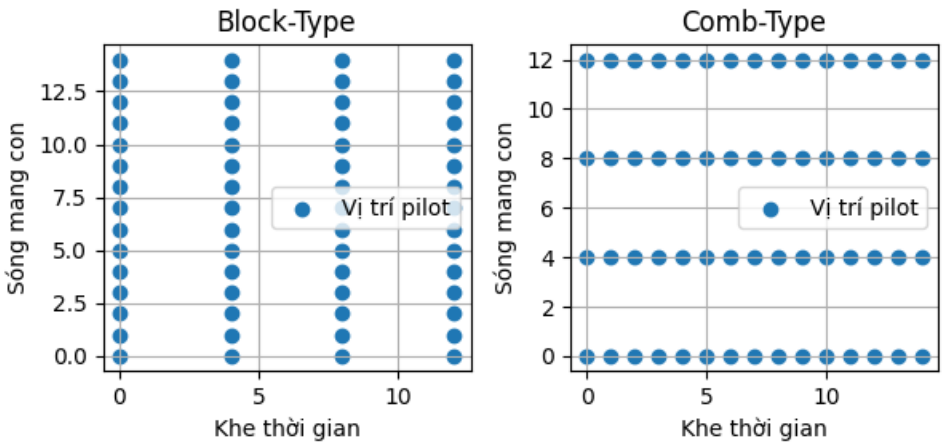
$\boldsymbol{H}$ là yếu tố cần được ước lượng chính xác để thiết bị thu có thể bù trừ các tác động của kênh và khôi phục lại $\boldsymbol{X}$ từ tín hiệu thu được $\boldsymbol{Y}$. Thông thường, kênh vô tuyến thay đổi theo thời gian và không gian, do đó việc ước lượng kênh đòi hỏi phải liên tục cập nhật trong suốt quá trình truyền dẫn.

2.2 Ký tự pilot

Ký tự pilot là các tín hiệu đã biết được chèn vào khung thời gian hoặc tần số của tín hiệu truyền để hỗ trợ ước lượng kênh. Các ký tự này được chèn vào với khoảng cách đã xác định trong cả miền thời gian và tần số (như trong hệ thống OFDM). Khi thiết bị thu nhận được các ký tự này, nó có thể sử dụng chúng để ước lượng các tham số của kênh tại những vị trí đã xác định, sau đó nội suy ra các vị trí khác.

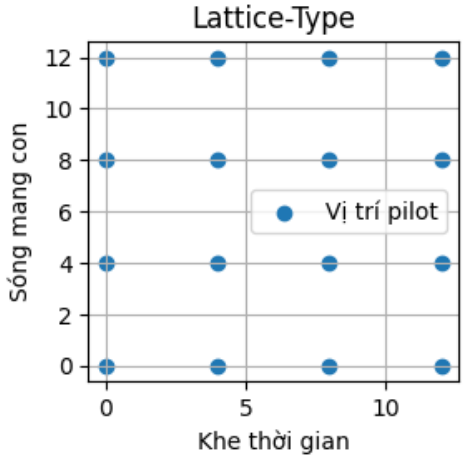
Các cấu trúc chèn ký tự pilot thông thường gồm:

- **Block-type**: Chèn pilot ở tất cả các subcarrier tại một vài khung thời gian.
- **Comb-type**: Chèn pilot vào một số subcarrier nhất định trong tất cả các khung thời gian.



Hình 1: Chèn pilot kiểu block-type và comb-type

- **Lattice-type**: Pilot được chèn vào theo mô hình hình học (như lưới) trong cả hai miền thời gian và tần số.



Hình 2: Chèn pilot kiểu lattice-type

2.3 Ước lượng kênh bằng LS

Phương pháp ước lượng Least Squares (LS) là một trong những phương pháp đơn giản và phổ biến nhất để ước lượng kênh trong hệ thống thông tin vô tuyến. Mặc dù độ chính xác của nó không cao bằng các phương pháp khác như MMSE, nhưng LS có ưu điểm về tính đơn giản và không yêu cầu thông tin thống kê của kênh, điều này làm cho nó trở nên hiệu quả trong các ứng dụng không đòi hỏi hiệu suất tối ưu hoặc có giới hạn về tài nguyên tính toán.

2.3.1 Nguyên lý của phương pháp LS

Ý tưởng cơ bản của phương pháp LS là tìm ra ma trận kênh $\boldsymbol{H}$ sao cho sai số giữa tín hiệu nhận được $\boldsymbol{Y}$ và tín hiệu truyền đi $\boldsymbol{X}$, sau khi đi qua kênh, là nhỏ nhất. Phương pháp này dựa trên việc tối thiểu hóa sai số bình phương giữa tín hiệu thu nhận được và tín hiệu dự đoán (dựa trên mô hình kênh truyền và các ký tự pilot đã biết).

Giả sử mô hình kênh truyền được biểu diễn dưới dạng:

$$\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N} \tag{2}$$

Trong đó:

- $\boldsymbol{Y}$: Tín hiệu nhận được tại thiết bị thu.
- $\boldsymbol{X}$: Tín hiệu đã truyền đi từ máy phát.
- $\boldsymbol{H}$: Ma trận kênh vô tuyến biểu diễn đặc tính của kênh (suy hao, trễ, tán xạ).
- $\boldsymbol{N}$: Nhiễu trắng Gaussian cộng (AWGN), mô tả nhiễu ảnh hưởng đến tín hiệu.

Phương pháp LS tìm H sao cho sai số bình phương giữa $\boldsymbol{Y}$ và $\boldsymbol{H} \cdot \boldsymbol{X}$ là nhỏ nhất, tức là:

$$\boldsymbol{H_{LS}} = \arg\min_{\boldsymbol{H}} \|\boldsymbol{Y} - \boldsymbol{H} \cdot \boldsymbol{X}\|^2 \tag{3}$$

2.3.2 Công thức ước lượng LS

Để giải bài toán tối thiểu hóa sai số bình phương, ta sẽ tính đạo hàm của hàm mục tiêu đối với $\boldsymbol{H}$ và đặt đạo hàm này bằng 0. Kết quả ta có được công thức ước lượng LS như sau:

$$\boldsymbol{H_{LS}} = \boldsymbol{Y} \cdot \boldsymbol{X}^{+} \tag{4}$$

Trong đó: $\boldsymbol{X}^+$ là **nghịch đảo giả Moore-Penrose** của ma trận $\boldsymbol{X}$. Nghịch đảo giả $\boldsymbol{X}^+$ thường được sử dụng khi $\boldsymbol{X}$ không phải là ma trận vuông hoặc không khả nghịch.

Nếu $\boldsymbol{X}$ là ma trận vuông và khả nghịch, công thức này đơn giản hơn:

$$\boldsymbol{H}_{LS} = \boldsymbol{Y} \cdot \boldsymbol{X}^{-1} \tag{5}$$

2.3.3 Ước lượng LS tại các vị trí pilot

Trong thực tế, tín hiệu pilot $\boldsymbol{X_p}$ được chèn vào một số vị trí đã biết trong khung thời gian-tần số để hỗ trợ ước lượng kênh. Tại các vị trí có ký tự pilot, tín hiệu nhận được có thể được biểu diễn dưới dạng:

$$\boldsymbol{Y_p} = \boldsymbol{H_p} \cdot \boldsymbol{X_p} + \boldsymbol{N_p} \tag{6}$$

Trong đó:

- Y_p là tín hiệu thu được tại các vị trí pilot,
- X_p là tín hiệu pilot đã truyền đi (đã biết),
- H_p là ma trận kênh tại các vị trí pilot cần được ước lượng,
- N_p là nhiễu tại các vị trí pilot.

Dựa trên công thức LS, ma trận kênh tại các vị trí pilot có thể được ước lượng như sau:

$$\boldsymbol{H_{LS}} = \frac{\boldsymbol{Y_p}}{\boldsymbol{X_p}} = \boldsymbol{Y_p} \cdot \boldsymbol{X_p^+} \tag{7}$$

2.3.4 Nội suy để ước lượng kênh tại các vị trí không phải pilot

Phương pháp LS chỉ ước lượng được ma trận kênh $\boldsymbol{H}$ tại các vị trí có ký tự pilot. Để tính toán giá trị kênh tại các vị trí không có ký tự pilot, các phương pháp **nội suy hai chiều** (2D interpolation) sẽ được sử dụng. Một số phương pháp nội suy phổ biến bao gồm:

- Nội suy tuyến tính (Linear interpolation),
- Nội suy spline,
- Nội suy bậc cao (Higher-order interpolation).

Nội suy giúp tạo ra một ma trận kênh ước lượng đầy đủ tại tất cả các vị trí trong khung thời gian và tần số, dựa trên thông tin từ các vị trí pilot.

3 Giải pháp ước lượng kênh MMSE, LS, và máy học

3.1 Ước lượng kênh bằng MMSE

Phương pháp Minimum Mean Square Error (MMSE) là một trong những phương pháp ước lượng kênh truyền tiên tiến, mang lại độ chính xác cao hơn so với Least Squares (LS) bằng cách sử dụng thông tin thống kê về kênh truyền và nhiễu. MMSE không chỉ giảm thiểu sai số giữa kênh ước lượng và kênh thực mà còn tối ưu hóa hiệu suất trong môi trường có nhiễu và fading mạnh.

3.1.1 Nguyên lý của phương pháp MMSE

MMSE dựa trên việc tối thiểu hóa trung bình bình phương sai số (MSE) giữa kênh truyền thực tế và kênh ước lượng. Điều này có nghĩa là MMSE không chỉ dựa vào tín hiệu đã nhận mà còn sử dụng thông tin thống kê về kênh truyền và nhiễu, từ đó đưa ra ước lượng tốt hơn so với LS. Cụ thể, MMSE cố gắng tối thiểu hóa sai số kỳ vọng giữa kênh thực và kênh ước lượng bằng cách tìm ma trận lọc tối ưu.

Giả sử mô hình kênh truyền trong hệ thống là:

$$\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N} \tag{8}$$

Trong đó:

- $\boldsymbol{Y}$: Tín hiệu nhận được tại thiết bị thu.
- $\boldsymbol{X}$: Tín hiệu đã truyền đi từ máy phát.
- $\boldsymbol{H}$: Ma trận kênh vô tuyến biểu diễn đặc tính của kênh (suy hao, trễ, tán xạ).
- $\boldsymbol{N}$: Nhiễu trắng Gaussian cộng (AWGN), mô tả nhiễu ảnh hưởng đến tín hiệu.

Phương pháp MMSE tìm cách tối thiểu hóa sai số kỳ vọng giữa kênh thực $\boldsymbol{H}$ và kênh ước lượng $\hat{\boldsymbol{H}}$, hay:

$$\epsilon = \boldsymbol{E}\left[\|\boldsymbol{H} - \hat{\boldsymbol{H}}\|^2\right] \tag{9}$$

3.1.2 Công thức MMSE

Giả sử tín hiệu nhận được tại các vị trí pilot là $\boldsymbol{Y_p}$ và tín hiệu pilot đã truyền là $\boldsymbol{X_p}$, phương trình mô tả tín hiệu sẽ là:

$$\boldsymbol{Y_p} = \boldsymbol{H_p} \cdot \boldsymbol{X_p} + \boldsymbol{N_p} \tag{10}$$

Phương pháp MMSE ước lượng kênh tại các vị trí pilot thông qua công thức:

$$\boldsymbol{H_{MMSE}} = \boldsymbol{R_{hy}} \cdot (\boldsymbol{R_{yy}} + \sigma_n^2 \boldsymbol{I})^{-1} \cdot \boldsymbol{Y_p} \tag{11}$$

Trong đó:

- $\boldsymbol{R_{hy}}$ là ma trận tương quan giữa kênh thực và tín hiệu nhận được (còn gọi là ma trận tương quan chéo giữa $\boldsymbol{H}$ và $\boldsymbol{Y}$),
- $\boldsymbol{R_{yy}}$ là ma trận tương quan của tín hiệu nhận được tại các vị trí pilot,
- σ_n^2 là phương sai của nhiễu cộng Gaussian,
- $\boldsymbol{I}$ là ma trận đơn vị.

Phương pháp này dựa vào việc mô tả thống kê của kênh truyền và tín hiệu nhận được. Nếu các ma trận tương quan này được ước lượng chính xác, MMSE sẽ cho kết quả ước lượng kênh chính xác hơn đáng kể so với LS.

3.1.3 Ước lượng kênh tại các vị trí không phải pilot

Để ước lượng kênh tại các vị trí không phải pilot, phương pháp nội suy có thể được áp dụng. Do MMSE đã khai thác thông tin thống kê của kênh, kết quả nội suy thường chính xác hơn so với nội suy từ ước lượng LS.

3.2 Ước lượng kênh bằng kĩ thuật học sâu

Phương pháp ước lượng kênh truyền vô tuyến dựa trên kỹ thuật học sâu (Deep Learning - DL) đã nổi lên như một giải pháp tiên tiến trong các hệ thống thông tin không dây, đặc biệt trong bối cảnh các hệ thống ngày càng phức tạp như MIMO hay OFDM. Các kỹ thuật học sâu có khả năng tự động học mô hình và xử lý các vấn đề phi tuyến mà các phương pháp truyền thống như LS hay MMSE gặp khó khăn.

3.2.1 Nguyên lý của ước lượng kênh dựa trên học sâu

Phương pháp ước lượng kênh truyền dựa trên học sâu sử dụng mạng nơ-ron nhân tạo (Artificial Neural Networks - ANN) để học cách biểu diễn và dự đoán kênh truyền từ dữ liệu nhận được, thay vì dựa trên các mô hình thống kê hoặc các công thức toán học cố định như trong MMSE hoặc LS. Mục tiêu chính là học mối quan hệ giữa tín hiệu nhận được và kênh truyền, từ đó ước lượng đặc tính kênh một cách chính xác và tự động.

Kênh truyền có thể được coi như một bản đồ hai chiều (2D) trong miền thời gian và tần số. Ý tưởng của DL là mô hình hóa kênh truyền như một hình ảnh hai chiều, với giá trị của kênh tại các vị trí pilot đã biết được xem như hình ảnh có độ phân giải thấp (LR - Low Resolution). Sau đó, mạng nơ-ron học cách "nâng cấp" hình ảnh đó thành hình ảnh có độ phân giải cao (HR - High Resolution), tức là ước lượng kênh tại các vị trí chưa biết.

3.2.2 Kiến trúc mạng học sâu cho ước lượng kênh

Phương pháp học sâu cho ước lượng kênh có thể sử dụng nhiều loại kiến trúc khác nhau, nhưng phổ biến nhất là Mạng nơ-ron tích chập (CNN - Convolutional Neural Networks). CNN đặc biệt phù hợp cho việc xử lý dữ liệu dạng lưới như hình ảnh hoặc dữ liệu thời gian-tần số trong các hệ thống thông tin không dây.

Mạng siêu phân giải ảnh (Super-Resolution CNN - SRCNN)

Mô hình **SRCNN** được áp dụng để nâng cao độ phân giải của các hình ảnh đại diện cho kênh truyền. Kênh truyền ở các vị trí pilot được coi như một ảnh có độ phân giải thấp, và mạng CNN sẽ thực hiện các phép biến đổi để dự đoán các giá trị kênh tại các vị trí chưa biết, giống như việc "phóng to" hình ảnh từ độ phân giải thấp lên độ phân giải cao.

SRCNN thường bao gồm các lớp sau:

- **Lớp tích chập đầu tiên**: Sử dụng các bộ lọc có kích thước lớn (ví dụ: 9×9) để học các đặc trưng ban đầu từ ảnh đầu vào.
- **Lớp tích chập giữa**: Dùng các bộ lọc nhỏ hơn (ví dụ: 1×1 hoặc 3×3) để học

các đặc trưng chi tiết hơn.

- **Lớp tích chập cuối**: Tái tạo lại ảnh có độ phân giải cao (ước lượng kênh đầy đủ) từ các đặc trưng đã học.

Mạng khử nhiễu ảnh (Denoising CNN - DnCNN)

Một thách thức lớn trong ước lượng kênh là sự hiện diện của nhiễu. DnCNN là một mô hình CNN được thiết kế để loại bỏ nhiễu khỏi hình ảnh. Trong bối cảnh ước lượng kênh, DnCNN được sử dụng để cải thiện chất lượng của kênh ước lượng sau khi mạng SRCNN đã ước lượng được các giá trị ban đầu, bằng cách loại bỏ các thành phần bị ảnh hưởng bởi nhiễu.

Mạng DnCNN bao gồm nhiều lớp tích chập với bộ lọc nhỏ (ví dụ: 3×3), giúp loại bỏ nhiễu từ tín hiệu đã được ước lượng, và sử dụng cơ chế học dựa trên residual learning (học phần dư) để tăng tốc quá trình hội tụ của mô hình.

3.2.3 Quy trình ước lượng kênh dựa trên học sâu

Các bước chính trong quy trình ước lượng kênh dựa trên học sâu:

Thu thập dữ liệu và tiền xử lý

- **Thu thập dữ liệu**: Dữ liệu bao gồm các tín hiệu nhận được tại máy thu cùng với các thông tin pilot đã biết. Các mẫu dữ liệu này sẽ được dùng để huấn luyện mô hình DL.
- **Chuẩn hóa dữ liệu**: Dữ liệu phải được chuẩn hóa để phù hợp với đầu vào của mạng nơ-ron. Ví dụ, dữ liệu có thể được chuẩn hóa về các phạm vi giá trị chuẩn để mô hình học dễ dàng hơn.

Huấn luyện mô hình học sâu

Mạng học sâu (SRCNN, DnCNN) được huấn luyện trên dữ liệu từ kênh truyền để học cách ước lượng các giá trị kênh dựa trên các mẫu tín hiệu nhận được. Quá trình huấn luyện thường bao gồm:

- **Xác định hàm mất mát**: Hàm mất mát thường được sử dụng là Mean Square Error (MSE) giữa kênh ước lượng và kênh thực, tức là:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{H_i} - \hat{\boldsymbol{H}}_{\boldsymbol{i}}\|^2 \tag{12}$$

Trong đó $\boldsymbol{H_i}$ là kênh thực và $\hat{\boldsymbol{H}}_{\boldsymbol{i}}$ là kênh ước lượng từ mạng nơ-ron.

- **Tối ưu hóa**: Các kỹ thuật tối ưu hóa như Stochastic Gradient Descent (SGD) hoặc Adam được sử dụng để cập nhật trọng số của mạng nhằm giảm hàm mất mát.

Kiểm thử và đánh giá

Sau khi huấn luyện, mô hình DL được kiểm thử trên một tập dữ liệu mới để đánh giá khả năng ước lượng kênh. Kết quả thường được so sánh với các phương pháp truyền thống như LS hoặc MMSE dựa trên các chỉ số như MSE hoặc Signal-to-Noise Ratio (SNR).

4 Chương trình Python mô phỏng ước lượng kênh truyền

File `models.py` chứa các hàm phục vụ cho việc ước lượng kênh truyền vô tuyến bằng kỹ thuật học sâu, với hai mô hình chính là SRCNN và DnCNN.

Hàm `psnr()` tính toán chỉ số PSNR (Peak Signal-to-Noise Ratio), dùng để đánh giá chất lượng hình ảnh tái tạo `target` so với ảnh gốc `ref`. Công thức tính PSNR dựa trên căn sai số bình phương trung bình (RMSE) giữa hai ảnh, sau đó chuyển đổi sang dB:

```
def psnr(target, ref):
    # assume RGB image
    target_data = np.array(target, dtype=float)
    ref_data = np.array(ref, dtype=float)

    diff = ref_data - target_data
    diff = diff.flatten('C')

    rmse = math.sqrt(np.mean(diff ** 2.))

    return 20 * math.log10(255. / rmse)
```

Hàm `interpolation(noisy, snr, number_of_pilot, interp)` thực hiện nội suy tín hiệu nhận được dựa trên số lượng pilot đã chọn và phương pháp nội suy (RBF hoặc spline). Hàm này tạo ra ảnh kênh truyền có độ phân giải cao bằng cách nội suy từ các giá trị pilot, sau đó trả về ảnh nội suy với kích thước tương ứng.

Hàm `SRCNN_model()` tạo và trả về mô hình Super-Resolution Convolutional Neural Network (SRCNN). Mô hình bao gồm 3 lớp tích chập:

- Lớp đầu tiên có 64 bộ lọc kích thước 9×9.
- Lớp thứ hai có 32 bộ lọc kích thước 1×1.
- Lớp cuối cùng có 1 bộ lọc kích thước 5×5.

Mô hình được biên dịch với hàm mất mát mean squared error (MSE) và được tối ưu bằng Adam.

```
def SRCNN_model():
    input_shape = (72, 14, 1)
    x = Input(shape=input_shape)
    c1 = Conv2D(64, (9, 9), activation="relu", padding="same",
       kernel_initializer="he_normal")(x)
    c2 = Conv2D(32, (1, 1), activation="relu", padding="same",
```

```
        kernel_initializer="he_normal")(c1)
    c3 = Conv2D(1, (5, 5), padding="same",
        kernel_initializer="he_normal")(c2)
    # c4 = Input(shape = input_shape)(c3)
    model = Model(inputs=x, outputs=c3)
    # compile
    adam = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999,
        epsilon=1e-8)
    model.compile(optimizer=adam, loss='mean_squared_error',
        metrics=['mean_squared_error'])
    return model
```

Hàm `SRCNN_train()` huấn luyện mô hình SRCNN trên dữ liệu kênh truyền. Đầu tiên, hàm này gọi hàm `SRCNN_model()` để tạo mô hình và sử dụng dữ liệu đầu vào để huấn luyện. Hàm này cũng sử dụng `ModelCheckpoint` để lưu mô hình với kết quả tốt nhất dựa trên validation loss.

```
def SRCNN_train(train_data, train_label, val_data, val_label,
    channel_model, num_pilots, SNR):
    srcnn_model = SRCNN_model()
    print(srcnn_model.summary())

    checkpoint = ModelCheckpoint("SRCNN_check.h5", monitor='val_loss',
        verbose=1, save_best_only=True,
                                 save_weights_only=False, mode='min')
    callbacks_list = [checkpoint]

    srcnn_model.fit(train_data, train_label, batch_size=128,
        validation_data=(val_data, val_label),
                    callbacks=callbacks_list, shuffle=True, epochs=300,
                        verbose=0)

    srcnn_model.save_weights("SRCNN_" + channel_model + "_" +
        str(num_pilots) + "_" + str(SNR) + ".h5")
```

Hàm `SRCNN_predict()` tải trọng số của mô hình SRCNN đã huấn luyện từ tệp và sử dụng mô hình để dự đoán dữ liệu đầu vào `input_data`.

```
def SRCNN_predict(input_data, channel_model, num_pilots, SNR):
    srcnn_model = SRCNN_model()
    srcnn_model.load_weights("SRCNN_" + channel_model + "_" +
        str(num_pilots) + "_" + str(SNR) + ".h5")
    predicted = srcnn_model.predict(input_data)
```

```
    return predicted
```

Hàm `DNCNN_model()` tạo và trả về mô hình Denoising Convolutional Neural Network (DnCNN). Mô hình bao gồm:

- Lớp tích chập đầu tiên với 64 bộ lọc 3×3 và hàm kích hoạt ReLU.
- 18 lớp tiếp theo gồm tích chập, batch normalization và ReLU.
- Lớp cuối cùng thực hiện phép trừ đầu vào với kết quả tích chập để loại bỏ nhiễu khỏi tín hiệu.

```
def DNCNN_model():
    inpt = Input(shape=(None, None, 1))
    # 1st layer, Conv+relu
    x = Conv2D(filters=64, kernel_size=(3, 3), strides=(1, 1),
        padding='same')(inpt)
    x = Activation('relu')(x)
    # 18 layers, Conv+BN+relu
    for i in range(18):
        x = Conv2D(filters=64, kernel_size=(3, 3), strides=(1, 1),
            padding='same')(x)
        x = BatchNormalization(axis=-1, epsilon=1e-3)(x)
        x = Activation('relu')(x)
        # last layer, Conv
    x = Conv2D(filters=1, kernel_size=(3, 3), strides=(1, 1),
        padding='same')(x)
    x = Subtract()([inpt, x]) # input - noise
    model = Model(inputs=inpt, outputs=x)
    adam = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999,
        epsilon=1e-8)
    model.compile(optimizer=adam, loss='mean_squared_error',
        metrics=['mean_squared_error'])
    return model
```

Hàm `DNCNN_train()` huấn luyện mô hình DnCNN trên dữ liệu kênh truyền và lưu lại mô hình tốt nhất. Quy trình tương tự như hàm huấn luyện cho SRCNN, với một checkpoint để lưu mô hình tốt nhất dựa trên validation loss.

```
def DNCNN_train(train_data, train_label, val_data, val_label,
    channel_model, num_pilots, SNR):
    dncnn_model = DNCNN_model()
    print(dncnn_model.summary())

    checkpoint = ModelCheckpoint("DNCNN_check.h5", monitor='val_loss',
```

```
        verbose=1, save_best_only=True,
                                save_weights_only=False, mode='min')
    callbacks_list = [checkpoint]

    dncnn_model.fit(train_data, train_label, batch_size=128,
        validation_data=(val_data, val_label),
                    callbacks=callbacks_list, shuffle=True, epochs=200,
                        verbose=0)
    dncnn_model.save_weights("DNCNN_" + channel_model + "_" +
        str(num_pilots) + "_" + str(SNR) + ".h5")
```

Hàm `DNCNN_predict()` tải mô hình DnCNN đã được huấn luyện và sử dụng để dự đoán trên dữ liệu đầu vào `input_data`.

```
def DNCNN_predict(input_data, channel_model, num_pilots, SNR):
    dncnn_model = DNCNN_model()
    dncnn_model.load_weights("DNCNN_" + channel_model + "_" +
        str(num_pilots) + "_" + str(SNR) + ".h5")
    predicted = dncnn_model.predict(input_data)
    return predicted
```