|  |  |
| --- | --- |
| **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM** | |
| **Độc lập – Tự do – Hạnh phúc** |

**LUẬN VĂN THẠC SĨ**

**Thông tin đề tài**

1. Họ và tên học viên: Ngô Thị Thanh Bình SHHV: CB160533

2. Chuyên ngành: Công nghệ thông tin Lớp: CNTT 2016B

3. Người hướng dẫn:

* TS. Phạm Văn Hải, Viện Công nghệ thông tin và Truyền thông, trường Đại học Bách khoa Hà Nội

4. Đơn vị: Viện Công nghệ Thông tin - Truyền thông.

5. Tên đề tài (tiếng Việt): Ứng dụng Học máy trong Kiểm thử phần mềm

6. Tên đề tài (tiếng Anh): Apply Machine Learning in Software Testing

# Lời cam đoan

# Lời cảm ơn

Mục lục

[Lời cam đoan 2](#_Toc15338882)

[Lời cảm ơn 3](#_Toc15338883)

[Danh sách hình vẽ 6](#_Toc15338884)

[Danh sách bảng 7](#_Toc15338885)

[Mở đầu 8](#_Toc15338886)

[Chương 1 Tổng quan về kiểm thử tự động và một số kiến thức cơ sở 10](#_Toc15338887)

[**1.** **Tổng quan về kiểm thử phần mềm** 10](#_Toc15338888)

[**1.1.** **Kiểm thử trong vòng đời phát triển phần mềm** 10](#_Toc15338889)

[**1.2.** **Kiểm thử tự động** 12](#_Toc15338890)

[**2.** **Tổng quan về học máy và mạng nơ-ron** 13](#_Toc15338891)

[**2.1.** **Giới thiệu học máy** 13](#_Toc15338892)

[**2.2.** **Mạng nơ ron** 14](#_Toc15338893)

[**2.3.** **Rừng ngẫu nhiên** 15](#_Toc15338894)

[**3.** **Tiếp cận phương pháp nghiên cứu cho bài toán kiểm thử phần mềm** 15](#_Toc15338895)

[**4.** **Tổng kết chương** 15](#_Toc15338896)

[Chương 2 Xây dựng hệ thống ứng dụng học máy vào kiểm thử phần mềm 16](#_Toc15338897)

[**1.** **Mô tả bài toán** 16](#_Toc15338898)

[**2.** **Mô hình đề xuất** 16](#_Toc15338899)

[**3.** **Xây dựng mô hình** 17](#_Toc15338900)

[**3.1.** **Xây dựng dữ liệu** 17](#_Toc15338901)

[**3.2.** **Mạng nơ ron** 18](#_Toc15338902)

[**3.3.** **Công cụ so sánh** 19](#_Toc15338903)

[**3.4.** **Dự đoán lỗi** 19](#_Toc15338904)

[**4.** **Ứng dụng** 20](#_Toc15338905)

[**5.** **Tổng kết chương** 21](#_Toc15338906)

[Chương 3 Cài đặt và thực nghiệm 21](#_Toc15338907)

[**1.** **Cài đặt chương trình** 21](#_Toc15338908)

[**1.1.** **Môi trường phát triển** 21](#_Toc15338909)

[**1.2.** **Thiết kế ứng dụng thử nghiệm** 22](#_Toc15338910)

[**1.3.** **Huấn luyện mạng nơ ron mô phỏng phần mềm** 25](#_Toc15338911)

[**2.** **Kết quả chương trình** 26](#_Toc15338912)

[**3.** **Tổng kết chương** 27](#_Toc15338913)

[Chương 4 Kết luận 27](#_Toc15338914)

[**1.** **Kết luận** 27](#_Toc15338915)

[**2.** **Định hướng phát triển** 27](#_Toc15338916)

[Tài liệu tham khảo 28](#_Toc15338917)

# Danh sách hình vẽ

[Hình 1.1 Quá trình học 14](file:///D:\kingslanding\ML_in_Testing\docs\LuanVan_NgoThiThanhBinh.docx#_Toc15338334)

[Hình 2.1 Mô hình kiến trúc tổng quát 16](#_Toc15338335)

[Hình 2.2 Tổng quát giai đoạn huấn luyện 17](#_Toc15338336)

[Hình 2.3 Tổng quát giai đoạn đánh giá 18](#_Toc15338337)

[Hình 3.1 Thuật toán phần mềm kiểm thử 22](#_Toc15338338)

[Hình 3.2 Quá trình huấn luyện mạng nơ ron 25](#_Toc15338339)

# Danh sách bảng

[Bảng 2.1 Phân loại các trường hợp đầu ra 19](#_Toc15338340)

[Bảng 2.2 Phân loại các kết luận 19](#_Toc15338341)

[Bảng 3.1 Đặc tả các yếu tố đầu vào đầu ra 23](#_Toc15338342)

[Bảng 3.2 Dữ liệu ví dụ 23](#_Toc15338343)

[Bảng 3.3 Các loại lỗi được đưa vào phần mềm 24](#_Toc15338344)

[Bảng 3.4 Tỉ lệ lỗi của mô hình dự đoán 26](#_Toc15338345)

# Mở đầu

**Lí do chọn đề tài**

Trong một thập kỉ trở lại đây ở ngành công nghiệp phần mềm, kiểm thử đóng một vai trò cực kì quan trọng và đã trở thành một quy trình riêng rẽ nhận được nhiều sự quan tâm [1]. Lý do của việc kiểm thử phần mềm đó là hiểu biết về chất lượng phần mềm hay khả năng chấp nhập và tìm hiểu các vấn đề [2]. Do sự phát triển cực kì nhanh của ngành công nghiệp phần mềm, nhiều ý kiến cho rằng kiểm thử đã trở thành khâu quan trọng và tốn kém thời gian nhất trong vòng đời phát triển phần mềm [3]. Hơn nữa, các kĩ thuật ứng dụng trong kiểm thử được xem là chưa theo kịp các xu hướng công nghệ hiện tại và cần nhiều hơn sự chú ý [4]. Không ngoài ngoại lệ, thực tế khâu này ở Việt Nam còn nhiều bất cập. Điển hình là quá trình kiểm thử thủ công tốn nhiều thời gian, chi phí mà hiệu quả không cao được như kì vọng. Nhân lực về kiểm thử cũng còn thấp cả về số lượng lẫn trình độ đào tạo. [5]

Dưới áp lực phải cho ra đời sản phẩm nhanh hơn, việc đảm bảo chất lượng lại càng gặp nhiều khó khăn. Làn sóng đầu tiên nổi lên trong trong các giải pháp kiểm thử thông minh đó là kiểm thử tự động [6], máy sẽ mô phỏng lại việc kiểm thử còn con người tập trung vào việc phát triển và điều khiểu nó.

Cùng với sự phát triển của các xu hướng công nghệ mới, một làn sóng mới nổi đó là ứng dụng xoay quanh trí tuệ nhân tạo và học máy [7]. Thay vì điều khiển bước một cho máy, con người hoàn toàn có thể chỉ cần quan tâm tới đầu vào/ đầu ra và không cần bỏ nhiều công sức theo sát quá trình máy thực hiện. Do đó, nghiên cứu luận văn đề cập tới vấn đề ứng dụng kỹ thuật thông minh, trợ giúp quá trình kiểm thử.

**Tính cấp thiết của đề tài**

Các công ty hiện nay đang đầu tư một lượng thời gian và công sức đáng kể vào phát triển và thiết kế sản phẩm phần mềm. Thực tế, khoảng 25% thời gian được dùng để sửa các lỗi cũng như đảm bảo việc phần mềm đáp ứng đúng yêu cầu đặt ra. [8] ở các công ty lớn, đặc biệt những nơi phải xây dựng những hệ thống cực kì phức tạp hoặc thừa kế lớn, rất khó để thay đổi cả hệ thống khi sản phẩm lớn dần [9], khi mà tất cả những tác động của việc thay đổi đều phải được kiểm thử đầy đủ.

Khi sản phẩm phần mềm lớn dần, phức tạp dần, các vấn đề liên quan tới kiểm thử cũng tăng tương tự [10], khiến các kĩ sư lập trình và kiểm thử cũng gặp nhiều thách thức hơn. Một khó khăn nữa là việc triển khai các mô hình kiểm thử. Mô hình này được thiết kế để kiểm tra xem đầu ra của một trường hợp kiểm thử có đúng với đầu ra dự đoán của chương trình [11]

Việc ứng dụng các kỹ thuật học máy vào kiểm thử phần mềm là cần thiết để tiết kiệm công sức kiểm thử tay cũng như nâng cao hiệu quả hoạt động kiểm thử. Kết quả từ ứng dụng dựa trên học máy cùng với nhu cầu thực tiễn, người dùng sẽ có thể đạt được mục tiêu của việc kiểm thử đó là tìm lỗi một cách có thể tin tưởng cao hơn.

**Mục đích nghiên cứu**

Mục đích chính của luận văn này, xét trong trường hợp ứng dụng thử, là nghiên cứu khả năng xây dựng một mô hình dự đoán trong đó, với đầu vào là phiên bản hiện tại của phần mềm, chính xác dự đoán các lỗi có thể xảy ra.

**Phạm vi nghiên cứu**

Phạm vi nghiên cứu của luận văn tập trung vào kiểm thử tự động, các đặc trưng của kiểm thử phần mềm và phương pháp trợ giúp kiểm thử tự động dựa trên các công nghệ học máy. Mô hình dự đoán lỗi dựa trên mạng nơ ron nhân tạo.

**Tóm tắt các nội dung chính trong luận văn**

Bố cục luận văn chia thành bốn chương:

Chương 1 trình bày kiến thức cơ sở về kiểm thử phần mềm, các kĩ thuật học máy phổ biến có thể ứng dụng trong ngành kiểm thử phần mềm. Cuối chương giới thiệu một số công trình liên quan và tiếp cận vấn đề. Chương 2 trình bày mô tả chi tiết các bước thực hiện được thực hiện trong luận văn để giải quyết bài toán. Chương 3 cài đặt thực nghiệm và phân tích kết quả. Cuối cùng là chương 4 kết luận về công trình nghiên cứu và ý tưởng phát triển thêm trong tương lai.

# Tổng quan về kiểm thử tự động và một số kiến thức cơ sở

1. **Tổng quan về kiểm thử phần mềm**
   1. **Kiểm thử trong vòng đời phát triển phần mềm**

Kiểm thử phần mềm là phân tích được thực hiện để đánh giá chất lượng của hệ thống phần mềm dưới dạng một bài kiểm tra [12]. Có rất nhiều kỹ thuật kiểm thử để đảm bảo chắc chắn rằng một bộ phận hay một hệ thống được kiểm tra sẽ đáp ứng đúng yêu cầu đặc tả. Rất nhiều các phương án khác nhau được sử dụng để chọn ra kỹ thuật kiểm thử hợp lí nhất. Điều này là do khối lượng kiểm thử là rất lớn trong khi thời gian và các chi phí khác là có hạn. Trong mô hình phát triển phần mềm, thời gian và tài nguyên cho kiểm thử là giới hạn. Mô hình phát triển phần mềm thác nước truyền thống [13] kiểm thử chỉ diễn ra khi hệ thống đã được thiết kế và triển khai xong toàn bộ. Tuy nhiên, trong mô hình phát triển nhanh agile ngày nay, kiểm thử diễn ra cùng giai đoạn với quá trình phát triển phần mềm.

Chương này sẽ giới thiệu các phương pháp kiểm thử khác nhau cùng giải thích. Chúng có thể sẽ được nhóm lại dựa theo bộ tiêu chí nhất định và một phương pháp có thể thuộc về nhiều nhóm. Mỗi nhóm sẽ có những lợi thế và bất lợi nhất định mà người kiểm thử phải xem xét và đánh giá kĩ nhằm tận dụng chúng tốt nhất có thể.

Đầu tiên, đứng từ phía người thiết kế các trường hợp kiểm thử, các phương pháp kiểm thử sẽ được chia làm hai loại chính là hộp trắng và hộp đen [14], ngoài ra còn có thêm hộp xám [15]. Thiết kết theo phương pháp hộp trắng cần có hiểu biết bên trong, cách hệ thống hoạt động. Ngược lại, thiết kế kiểm thử theo phương pháp nhóm hộp đen sẽ đánh giá tính năng của hệ thống mà không có hiểu biết gì về cách hoạt động bên trong hay mã triển khai. Hộp xám là phương pháp kết hợp khi hiểu biết về phần mềm được dùng để thiết kế các trường hợp kiểm thử còn việc thực hiện kiểm thử lại ở mức độ người dùng.

Tiếp tục, các phương pháp kiểm thử còn được phân chia theo mức độ. Thấp nhất là kiểm thử đơn vị [16] mà theo đó từng phần mã nguồn riêng rẽ cùng với dữ liệu đi kèm, thủ tục sử dụng và vận hành được kiểm thử để xác định là chúng hoạt động đúng. Những bài kiểm thử này chủ yếu viết bởi người lập trình viên sau khi hoặc thậm chí trước cả khi bắt đầu viết mã nguồn phần mềm [17] nhằm khuyến khích việc thiết kế kiến trúc phần mềm đơn giản hơn. Kiểm thử hộp trắng cũng liên quan trực tiếp tới mức độ kiểm thử này vì sự tương đồng về trong tính chất. Mức độ thứ hai là kiểm thử kết hợp [18] là kiểm thử sự kết nối giữa các thành phần, tương tác tới các phần khác nhau trong một hệ thống đang vận hành. Tiếp theo là kiểm thử hệ thống đề cập tới hành vi của hệ thống được định nghĩa bởi phạm vi phát triển phần mềm [18]. Cuối cùng là kiểm thử chấp nhận, được thực hiện từ phía khách hàng cũng tại chính môi trường mà nó sẽ được vận hành. Trong đó, các trường hợp kiểm thử cũng sẽ chỉ gồm những điểm quan trọng từ điểm nhìn của khách hàng.

Các kĩ thuật kiểm thử còn có thể được chia theo công cụ, phần mềm hỗ trợ. Kiểm thử thủ công là các trường hợp việc kiểm thử được thực hiện mà không có sự hỗ trợ của phần mềm tự động nào. Người kiểm thử sẽ đi qua tất cả các bước trong kế hoạch kiểm thử để đảm bảo tất cả các chức năng của hệ thống có kết quả đúng theo yêu cầu. Các công đoạn này đều có thể được tự động hóa bằng phần mềm. Tức là, một phần mềm tự động được điều khiển để thực hiện theo trường hợp kiểm thử và so sánh với kết quả đã định sẵn. [19] Các cách nhìn khác nhau về phân loại kiểm thử đều có thể được kết hợp lại và trong các công ty phần mềm ngày nay, nhiều phương án kiểm thử khác nhau đang được dùng để đảm bảo chất lượng. Và xu hướng hiện nay đó là đơn giản hóa việc kiểm thử tự động để làm đòn bẩy cho việc đầu tư vào kiểm thử thủ công.

Bất kể là hình thức kiểm thử nào, để có hiệu quả, cần phải chú ý tới các điểm sau:

* **Yêu cầu:** Bất cứ đơn hàng phần mềm nào đều bắt đầu với một bộ yêu cầu, chính là đầu ra mà phần mềm cần kiểm thử phải đảm bảo. Mỗi yêu cầu phải có ít nhất một trường hợp kiểm thử được bao gồm trong kế hoạch kiểm thử [20]
* **Độ bao phủ:** Khái niệm này nói tới tỉ lệ mã nguồn mà bộ kiểm thử có thể xử lí. Đây là vấn đề đáng ngại khi lập trình viên triển khai kiểm thử đơn vị bởi thực sự để đạt độ bao phủ 100% là không thể. Thật sự, đạt tỉ lệ bao phủ 100% có nghĩa là lập trình viên thực hiện những bài kiểm thử kém chất lượng chỉ để đạt đủ tỉ lệ [21].
* **Thời gian và kinh phí:** Mất bao nhiêu thời gian để dành cho việc kiểm thử, bao nhiêu tài nguyên (phần cứng, công cụ, nhân sự,…) phục vụ quá trình kiểm thử.
  1. **Kiểm thử tự động**

Kiểm thử phần mềm về bản chất có thể xem là sự lặp đi lặp lại. Một danh sách các bước cần thực hiện, kiểm tra đầu ra, đưa hệ thống về trạng thái ban đầu và thực hiện lại các bước từ đầu. Các nhân viên kiểm thử cùng triển khai trên một hệ thống, làm các bước giống hoặc gần giống nhau. Như vậy, kiểm thử là một hoạt động tương đối chậm và tốn kém. Sự thiếu khách quan của người kiểm thử có thể ảnh hưởng tới các bước quan trọng trong quy trình kiểm thử. Tuy nhiên, trong kiểm thử tự động, những điểm vừa liệt kê khó có thể xảy ra

* **Tăng năng xuất:** Lý tưởng mà nói, một bộ trường hợp kiểm thử tự động nên được chạy mà không có sự can thiệp của con người. Lấy ví dụ, sau khi có bản phần mềm cần kiểm thử và trước khi ra về, người ta bắt đầu bộ kiểm thử và sáng hôm sau đến chỉ cần phải kiểm tra báo cáo. Như vậy đã tiết kiệm được rất nhiều tài nguyên khi để việc kiểm thử chạy tự động ban đêm không cần người giám sát. [22]

Kiểm thử tự động có thể làm ngắn lại vòng đời phát hành sản phẩm. Những lỗi do thay đổi trong hệ thống sẽ sớm được phát hiện sớm trong quy trình phát triển phần mềm, tiết kiệm công sức và tiền bạc [23] Hơn nữa, kiểm thử tự động có thể xử lí lượng lớn dữ liệu so với kiểm thử thủ công, ví dụ cần kiểm thử trên bộ dữ liệu gồm hàng ngàn bản ghi sẽ gặp khó khăn nếu làm thủ công trong khi khá là đơn giản nếu thực hiện tự động.

* **Tăng độ tin cậy:** Điều này thể hiện ở việc kiểm thử tự động được chạy chính xác giống hệt nhau mỗi lần thực hiện. Một hoạt động kiểm thử thủ công chắc chắc khó tránh khỏi các lỗi ở con người, nhất là ở các trường hợp lặp đi lặp lại, cần chuẩn bị nhiều, hoặc cần chú ý tới các bước phụ, nhỏ, chi tiết. Hơn nữa, việc để kiểm thử chạy tự động giải thoát người kĩ sư khỏi những việc lặp đi lặp lại mà tập trung hơn vào nghiên cứu hệ thống, thiết kế kế hoạch kiểm thử hiệu quả hơn. [23]
* **Tăng độ bao phủ:** Kiểm thử tự động chạy rất tương đối nhanh. Nhiều trường hợp kiểm thử được thực thi trong thời gian ngắn hơn sẽ dẫn đến việc có thể chạy nhiều trường hợp kiểm thử hơn, độ bao phủ phần mềm cao hơn.

1. **Tổng quan về học máy và mạng nơ-ron**
   1. **Giới thiệu học máy**

Học máy là kĩ thuật chính được sử dụng trong luận văn này và chính là quy trình để tìm ra các đặc tính về phần mềm thông qua dữ liệu [24]. Có những mẫu dữ liệu là khó thấy với con người nhưng lại dễ thấy khi cho đi qua các kĩ thuật học máy. Có rất nhiều loại phương pháp tiếp cận học máy khác nhau; chúng được phân ra thành ba nhóm chính dựa theo dữ liệu: có thể kể đến, học có giám sát, không giám sát và học tăng cường [25]. Trong học có giám sát, ta có một bộ đầu vào và đầu ra được gọi là dữ liệu huấn luyện. Thuật toán thuộc học có giám sát phân tích dữ liệu huấn luyện và thực hiện suy luận, có thể sử dụng để đối chiếu với dữ liệu mới [26]. Học không giám sát có thể hiểu là không có nhãn huấn luyện, thuật toán sẽ không được giám sát và sẽ phải tự tìm ra cấu trúc dựa trên dữ liệu đưa vào [27]. Cuối cùng, học tăng cường là cách một chương trình máy tính tương tác với hệ thống thay đổi thường xuyên và cố gắng học từ đó. Với hình thức này, máy sẽ học từ việc thử rồi gặp lỗi lặp đi lặp lại [28].

Trong luận án này chủ yếu đi tìm khả năng tạo một mô hình dự đoán có thể đem đến một phương án đo đạc được để quyết định lỗi của phần mềm. Dữ liệu cho mục đích này sẽ phải có nhãn – tức là ta đã biết đầu ra của phần mềm mà ta cần kiểm thử. Theo đó, việc tìm hiểu sâu về các thuật toán học có giám sát là cần thiết. Các thuật toán thuộc nhóm này, với dữ liệu huấn luyện có nhãn, sẽ cho ta một mô hình dự đoán phù hợp, thể hiện trong Hình 1.1.

Hình 1.1 Quá trình học

Một số thuật toán thường gặp nhất trong nhóm thuật toán cho học có giám sát là: Máy vectơ hỗ trợ, mạng nơ ron, hồi quy logistic, naive bayes, rừng ngẫu nhiên và cây quyết định [29]. Một số nghiên cứu cũng chỉ ra những hữu ích và vấn đề khi ứng dụng học máy vào kiểm thử. Theo đó, mô hình kiểm thử là khó tự động hóa xong học máy có thể là một phẫn hữu dụng trong quá trình này, đặc biệt trong hệ thống thay đổi thường xuyên. Tính chính xác của dự đoán đưa ra bởi thuật toán không phải là thước đo duy nhất để lựa chọn phương pháp học máy mà cả là sự đơn giản của mô hình cũng quan hệ mật thiết [24].

Hai phương pháp học máy được sử dụng trong luận văn này là mạng nơ ron và rừng ngẫu nhiên, sẽ được bàn tới kĩ hơn trong các phần tiếp sau.

* 1. **Mạng nơ ron**

Mạng nơ ron là mô hình học máy ít nhiều dựa trên sự khám phá về nơ ron sinh học trong khoa học về thần kinh con người [29]. Một đơn vị trong mạng được gọi là một nơ ron, và nhận một số lượng tín hiệu vào nhất định. Các tín hiệu vào có thể bị biến đổi trọng số tại khớp nối. Các nơ ron sẽ có một bộ hàm tính toán tín hiệu đầu ra từ tín hiệu đầu vào. Về bản chất mạng này là tập hợp các hàm tính, không trực tiếp có từ dữ liệu, mà sẽ được tạo ra thông qua huấn luyện sử dụng một thuật toán học máy. Trong luận văn này, mạng nơ ron suy luận tiến sẽ được sử dụng, trong đó thông tin sẽ được truyền một chiều, từ tín hiệu vào tới tín hiệu ra.

* 1. **Rừng ngẫu nhiên**

Một mô hình phổ biến khác nữa đó chính là rừng ngẫu nhiên. Đây thực chất là một mô hình tổng hợp, có nghĩa là đầu ra ở các mô hình đơn giản hơn được sử dụng để đưa ra kết quả cuối cùng [30]. “Rừng ngẫu nhiên” này sẽ bao gồm một bộ cây hồi quy và phân loại, với đầu vào, đi qua các nhánh nhị phân tới ngọn cây. Mỗi lần rẽ nhánh sẽ chia cây ra thành các phần khác nhau, dựa theo một hoặc nhiều đầu vào, và ngọn cây cuối cùng sẽ là kết quả dự đoán. Kết quả của cả mô hình cuối cùng sẽ là kết quả trung bình từ mỗi cây.

1. **Tiếp cận phương pháp nghiên cứu cho bài toán kiểm thử phần mềm**

Xu hướng hiện nay trong kiểm thử phần mềm đó là kiểm thử tự động mọi thứ. Tuy nhiên, với sự phát triển của trí tuệ nhân tạo, thay vì con người tự tạo ra tất cả quá trình kiểm thử tự động, máy sẽ thay người viết và vận hành, đồng thời tiếp tục phát triển qua sự bổ sung của con người.

Một số tình huống mà phương pháp tiếp cận bằng trí tuệ nhân tạo sẽ giúp triệt tiêu những bất cập trong kiểm thử phần mềm:

* AI học về cách hoạt động, các lỗi thường gặp của phần mềm
* AI nhận biết sự thay đổi của phần mềm qua thời gian
* AI phân tích code và dự đoán các trường hợp kiểm thử

1. **Tổng kết chương**

# Xây dựng hệ thống ứng dụng học máy vào kiểm thử phần mềm

1. **Mô tả bài toán**

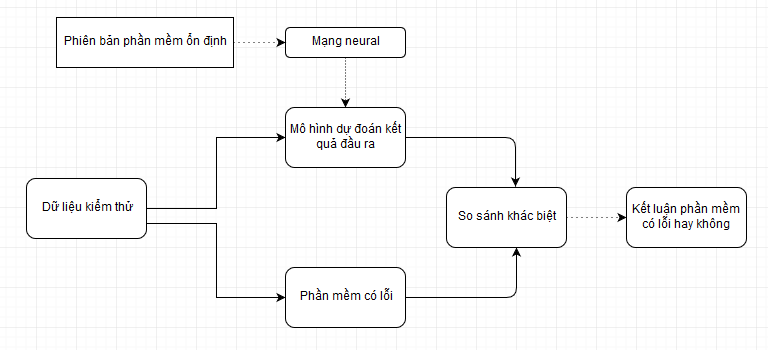
Khi thực hiện việc kiểm thử phần mềm, cơ bản nhất người ta cần xác định hai thứ: kết quả mong muốn và kết quả thực chạy trên phần mềm. Việc so sánh kết quả mong đợi với kết quả thực tế sẽ giúp phát hiện lỗi trên phần mềm.

Kết quả mong muốn này có được từ đặc tả yêu cầu phần mềm hoặc từ phiên bản chạy ổn định và đáng tin của phần mềm. Tuy nhiên điều này không phải lúc nào cũng dễ dàng. Trường hợp đầu tiên, đặc tả phần mềm thường sẽ chỉ bao gồm các yêu cầu mang tính ngữ nghĩa nhiều hơn là dữ liệu và kết quả cụ thể. Trường hợp thứ hai, phiên bản phần mềm đáng tin có thể không được giữ lại hoặc không sẵn sàng khi thay đổi môi trường, nâng cấp hệ thống,…

Do đó, bài toán ở đây là làm thế nào xây dựng một ứng dụng tự động tạo kết quả mong đợi cho phần mềm và các kết quả này cần có độ chính xác cao, từ đây sẽ gọi là “mô hình dự đoán.” Từ đó trợ giúp đánh giá hành vi của phần mềm có lỗi hay không, trợ giúp việc xây dựng các trường hợp kiểm thử nói riêng và chất lượng phần mềm nói chung.

1. **Mô hình đề xuất**

Kiến trúc ứng dụng học máy xây dựng mô hình kiểm thử phẩn mềm được có thể được xây dựng như hình sau:



Hình 2.1 Mô hình kiến trúc tổng quát

**Đầu vào**: dữ liệu kiểm thử phần mềm được chuẩn hóa để chạy trên cả mô hình dự đoán và phần mềm cần kiểm thử

**Đầu ra**: độ khác biệt của kết quả trên phần mềm cần kiểm thử so với kết quả mô hình, trong đó:

* Kết quả khác biệt lớn: kết luận phần mềm có lỗi
* Kết quả tương đồng lớn: kết luận phần mềm không có lỗi
* Kết quả không tỏ rõ khác biệt hay tương đồng rõ rệt: kết luận mô hình không đáng tin và cần cải tiến.

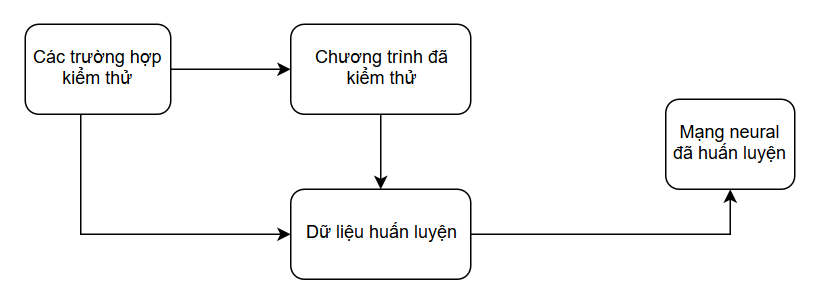
Mô hình mạng nơ ron đa lớp đã chứng tỏ khả năng giải quyết các bài toán thông dụng đến phức tạp một cách hiệu quả. Theo đó, phương pháp này được chọn để xây dựng mô hình dự đoán kết quả phần mềm cần kiểm thử.

1. **Xây dựng mô hình**
   1. **Xây dựng dữ liệu**

Để có được mô hình kiểm thử phần mềm như kiến trúc vừa trình bày ở trên, đầu vào, đầu ra và định nghĩa nhãn phải được xác định rõ ràng. Tùy thuộc vào từng bài toán kiểm thử cho phần mềm nhưng để thích hợp với mô hình dự đoán xây dựng từ mạng nơ ron, dữ liệu cần phải được chuẩn hóa về dạng vector. Đầu vào, đầu ra của phần mềm cần phải đưa về các dạng có thể tính toán và đo lường được.

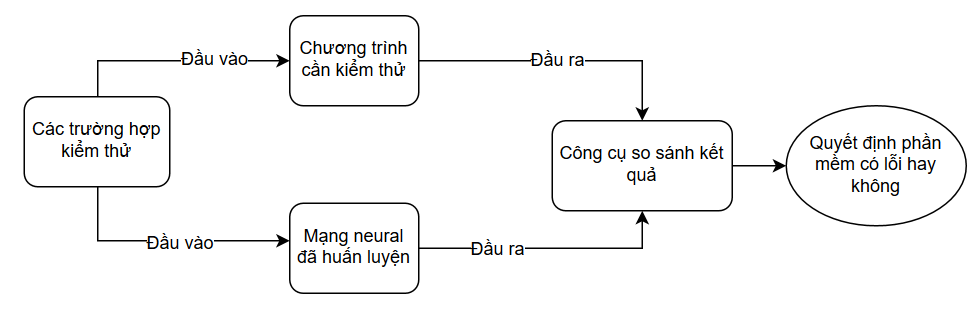
* 1. **Mạng nơ ron**

Giai đoạn huấn luyện: Các vector đầu vào (trường hợp kiểm thử) được lựa chọn ngẫu nhiên, tương ứng với đặc tả yêu cầu của phần mềm cần kiểm thử. Mỗi đầu vào này được đưa vào phần mềm có kết quả chạy ổn định và đã được kiểm thử để có được đầu ra tương ứng. Bộ đầu vào – đầu ra này được sử dụng để huấn luyện mạng nơ ron.



Hình 2.2 Tổng quát giai đoạn huấn luyện

Giai đoạn kiểm tra: Vẫn là bộ dữ liệu tương tự như ở giai đoạn huấn luyện nhưng lấy từ phiên bản mới hơn của phần mềm để kiểm tra kết quả của mạng nơ ron. Kiểm tra kết quả của mạng nơ ron trên bộ dữ liệu này nếu cho sai số dự đoán thấp đạt mong đợi, có thể coi đã thành công tạo mô hình và đưa mô hình này vào sử dụng trong kiểm thử phần mềm.



Hình 2.3 Tổng quát giai đoạn đánh giá

* 1. **Công cụ so sánh**

Công cụ so sánh này được coi là một phương pháp độc lập để so sánh kết quả từ mạng nơ ron và kết quả từ phiên bản phần mềm có lỗi. Việc so sánh này đương nhiên sẽ được thiết lập tự động để tránh kết quả bị ảnh hưởng bởi các yếu tố bên ngoài. Đồng thời, nó cũng thay thế những người kiểm thử thủ công có ý kiến chủ quan dựa trên kiến thức đã có về phần mềm.

Công cụ này về cơ bản sẽ đo khoảng cách tuyệt đối giữa kết quả của mô hình mạng nơ ron và kết quả phần mềm có lỗi với cùng một đầu vào. Khoảng cách tuyệt đối này sẽ được giới hạn giá trị trong đoạn [0.0;1.0] và được dùng để xác định kết quả từ phần mềm là đúng hay lỗi.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Đầu ra phần mềm cần kiểm thử | |
|  |  | Đúng | Sai |
| Đầu ra mạng nơ ron | Đúng | 1 | 2 |
| Sai | 4 | 3 |

Bảng 2.1 Phân loại các trường hợp đầu ra

Bảng 1 thể hiện sáu trường hợp tương ứng với các khả năng đầu ra của phần mềm và mạng nơ ron. Có các khả năng như vậy là bởi mạng nơ ron bản chất là mô phỏng của phần mềm thực sự nên chắc chắn không thể tránh khỏi một số ít kết quả sẽ không đúng. Về phía đầu ra của phần mềm cần kiểm thử đương nhiên sẽ có kết quả sai. Ví dụ nếu đầu ra của phần mềm là sai nhưng mạng nơ ron lại cho là đúng ta có trường hợp 2. Tương tự như vậy, các trường hợp còn lại cũng sẽ dẫn tới các kết luận tương ứng. Các trường hợp quan trọng cần quan tâm là 2, 3 khi phần mềm kiểm thử cho kết quả sai. Ngoài ra cần lưu ý hai bộ trường hợp 1 và 3, 2 và 4 khi công cụ so sánh sẽ cho ra kết quả giống nhau.

* 1. **Dự đoán lỗi**

Từ công cụ so sánh với việc chọn ngưỡng giới hạn so sánh khoảng cách tuyệt đối sẽ giúp kết luận phần mềm kiểm thử có lỗi hay không. Tương tự như trên ta lại có một bảng phân loại các trường hợp có thể xảy ra:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Phần mềm cần kiểm thử | |
|  |  | Có lỗi | Không lỗi |
| Mô hình dự đoán | Có lỗi | 1 | 2 |
| Không lỗi | 4 | 3 |
| Không chắc | 5 | 6 |

Bảng 2.2 Phân loại các kết luận

Có thêm kết quả không chắc nữa là do ngưỡng so sánh kết quả tuyệt đối giữa đầu ra mạng nơ ron với đầu ra phần mềm. Nếu so sánh đầu ra mạng nơ ron và đầu ra phần mềm cần kiểm thử rơi vào ngưỡng không quá tương đồng cũng không quá khác biệt, mô hình dự đoán sẽ không kết luận được là đúng hay sai.

Đây sẽ là công cụ chính để đánh giá độ tin tưởng cũng như cải thiện mô hình dự đoán lỗi. Do tính chất tích lợp lỗi (error convergence) của mạng nơ ron, khó đạt được một kết quả hoàn hảo xong hoàn toàn có thể tối ưu để đạt được mô hình có độ đáng tin nhất định.

1. **Ứng dụng**

Thử nghiệm trên ứng dụng phê duyệt kết quả tín dụng

Thử nghiệm này sẽ trực tiếp triển khai mô hình đề xuất trên và được chia làm ba phần, đầu tiên là phần mềm mẫu, hai là mạng nơ ron, và ba là công cụ so sánh và kết luận kết quả kiểm thử. Đặc tả yêu cầu cho phần mềm phê duyệt kết quả tín dụng được sử dụng để cung cấp đầu vào và đầu ra cũng như cài lỗi. Việc thiết kế mạng nơ ron cũng phụ thuộc vào kiểu dữ liệu đầu vào đầu ra này. Hình 1 và hình 2 diễn tả quy trình thử nghiệm sẽ diễn ra. Trong giai đoạn huấn luyện, dữ liệu đầu vào dùng cho mạng nơ ron và chương trình đã kiểm thử sẽ được tạo ra ngẫu nhiên. Trong giai đoạn đánh giá, một phần mềm có lỗi sẽ được tạo ra bằng cách thay đổi đoạn mã gốc trên phần mềm ban đầu.

*Thiết kế phần mềm phê duyệt kết quả tín dụng*

Phần mềm được sử dụng trong thử nghiệm là một phần mềm phê duyệt tín dụng đơn giản. Phần mềm này sẽ đọc các thông tin tổng hợp về khách hàng cần duyệt tín dụng và đưa ra hai kết luận. một là có duyệt tín dụng cho khách không, thứ hai là nếu duyệt thì khách hàng được cấp mức tín dụng bao nhiêu. Lập trình viên sẽ dựa theo các ràng buộc khác nhau đối với đầu vào để thiết kế thuật toán bám sát các ràng buộc đó. Phần mềm này có thể coi là có thể đại diện mức rộng và đủ, khi mà đầu ra chắc chắn sẽ phụ thuộc, có thể không phải tất cả, nhưng phần lớn ở đầu vào. Dữ liệu huấn luyện được tạo ra ngẫu nhiên sử dụng đặc tả phần mềm với mô tả chi tiết cho từng yếu tố ảnh hưởng tới đầu ra.

*Thiết kế mạng nơ ron*

Mạng nơ ron sử dụng trong thự nghiệm này sẽ dùng trực tiếp dữ liệu đầu vào của phần mềm phê duyệt tín dụng. Các đầu vào này đều được chuẩn hóa về dạng số. Có hai đầu ra thuộc hai loại khác nhau nên sẽ phải được xử lí khác nhau. Đầu ra cho việc có duyệt tín dụng hay không sẽ thuộc dạng hai lớp (0 và 1) còn đầu ra cho mức tín dụng là liên tục nhưng sẽ được xử lí để gom thành 10 nhãn (0 tới 9) và chuẩn hóa dựa trên mức cao nhất và thấp nhất.

*Thiết kế công cụ so sánh và kết luận*

Khi bắt đầu quá trình so sánh và kết luận, các lỗi sẽ lần lượt đưa vào phần mềm ban đầu. Các lỗi này có thể gây ảnh hưởng từ không đáng kể tới nghiêm trọng lên kết quả của phần mềm. Lúc này, công cụ so sánh sẽ mô phỏng lại quá trình kiểm thử phần mềm và kết luận phần mềm có lỗi hay không dựa trên kết quả chạy trên phần mềm đã bị sửa và kết quả từ mạng nơ ron. Các trường hợp kiểm thử này đều là độc lập không liên quan hay phụ thuộc lẫn nhau.

1. **Tổng kết chương**

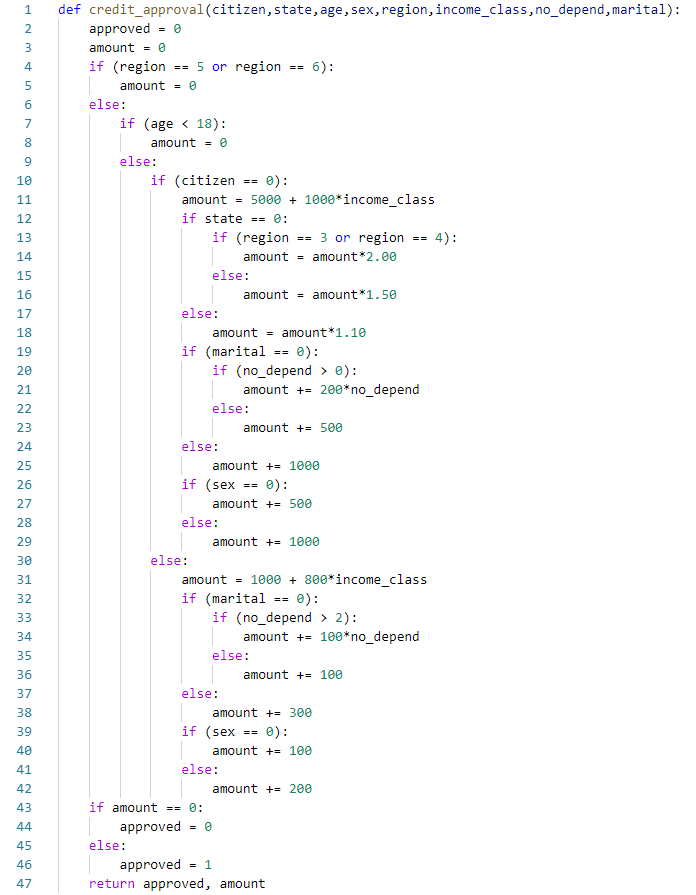
# Cài đặt và thực nghiệm

1. **Cài đặt chương trình**
   1. **Môi trường phát triển**

Mô hình mạng nơ ron được phát triển trong luận văn sử dụng trên môi trường:

* Ngôn ngữ lập trình: Python 3
* Thư viện phát triển mạng nơ ron: tensorflow, keras
* Chương trình chạy code: Jupyter Notebook
  1. **Thiết kế ứng dụng thử nghiệm**

Mô tả chi tiết thuật toán cho ứng dụng được thể hiện trong Hình 4 cùng chi tiết biến đầu vào, đầu ra, kiểu dữ liệu được thể hiện trong Bảng 3. Bảng 4 cho thấy dữ liệu mẫu.



Hình 3.1 Thuật toán phần mềm kiểm thử

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Kiểu dữ liệu** | **Kiểu yếu tố** | **Diễn giải** |
| Mã khách hàng | Integer | Đầu vào |  |
| Quốc tịch | Integer | Đầu vào | 0: Mỹ  1: Khác |
| Bang | Integer | Đầu vào | 0: Florida  1: Khác |
| Tuổi | Integer | Đầu vào | 1-100 |
| Giới tính | Integer | Đầu vào | 0: Nữ  1: Nam |
| Vùng | Integer | Đầu vào | 0-6 cho các vùng ở Mỹ |
| Mức thu nhập | Integer | Đầu vào | 0: Thu nhập năm dưới 10k USD  1: Thu nhập năm >= 10k USD  2: Thu nhập năm >= 25k USD  3: Thu nhập năm >= 50k USD |
| Số người phụ thuộc | Integer | Đầu vào | 0-4 |
| Tình trạng hôn nhân | Integer | Đầu vào | 0: Độc thân  1: Đã kết hôn |
| Duyệt cho vay | Integer | Đầu ra | 0: Không  1: Có |
| Mức duyệt cho vay | Integer | Đầu ra | >= 0 |

Bảng 3.1 Đặc tả các yếu tố đầu vào đầu ra

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| citizenship | state | age | sex | region | income\_class | no\_depend | marital | approved | amount |
| 1 | 1 | 95 | 1 | 2 | 0 | 1 | 0 | 1 | 1300 |
| 1 | 1 | 82 | 1 | 5 | 3 | 1 | 0 | 0 | 0 |
| 1 | 1 | 21 | 0 | 0 | 0 | 4 | 0 | 1 | 1500 |
| 1 | 0 | 7 | 1 | 6 | 2 | 2 | 0 | 0 | 0 |
| 1 | 0 | 81 | 0 | 2 | 2 | 1 | 0 | 1 | 2800 |
| 0 | 0 | 7 | 0 | 3 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 7 | 0 | 2 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 11 | 0 | 3 | 3 | 0 | 0 | 0 | 0 |
| 1 | 1 | 25 | 0 | 0 | 1 | 0 | 1 | 1 | 2200 |

Bảng 3.2 Dữ liệu ví dụ

Lấy ví dụ, khách hàng đầu tiên trong bảng, là người Mỹ sống ở bang Florida, Nam giới 95 tuổi đến từ vùng 2 ở Mỹ, thuộc mức thu nhập 0 (dưới 10 000 USD/năm) với 1 người phụ thuộc tình trạng hôn nhân là độc thân. Khách này được duyệt tín dụng với hạn mức là 1300 USD. Vì mạng nơ ron chỉ có thể nhận đầu vào dưới dạng số, các đầu vào có tính phân loại (quốc tịch, bang,…) được chuyển sang dạng số.

Hình 4 thể hiện rõ ràng chi tiết cấu trúc logic của chương trình duyệt tín dụng này. Từ đó có thể đoán được các loại lỗi có thể được đưa vào chương trình này. Cấu trúc này gồm một loạt lớp điều kiện. Nó cho phép kiểm tra hiệu ứng sẽ xảy ra với các lỗi đưa vào theo nhiều khả năng. Các loại lỗi này bao gồm những thay đổi nhỏ trong các lớp điều kiện. Có thể là thay đổi phép so sánh, thay đổi giá trị dùng trong điều kiện. Các thay đổi là độc lập và được thực hiện riêng lẻ chứ không đồng thời. Do đó, việc phân tích đầu ra cũng được thực hiện độc lập. Bảng 5 thể hiện các loại lỗi sẽ được đưa vào phần mềm.



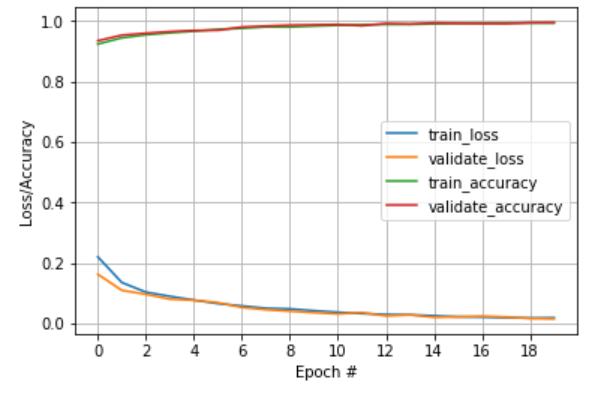
Bảng 3.3 Các loại lỗi được đưa vào phần mềm

* 1. **Huấn luyện mạng nơ ron mô phỏng phần mềm**

Cấu trúc mạng nơ ron được xây dựng:

* Đầu vào: 8 (Tương ứng 8 biến phần mềm)
* Đầu ra: 10 (Tương ứng 10 nhãn)
* Cấu trúc mạng: 2 lớp ẩn với lần lượt 300 và 150 nút
* Tốc độ học: 0,001
* Số epoch: 20
* Số batch: 50

Hình 5 thể hiện kết quả hàm lỗi và độ chính xác trong quá trình huấn luyện mạng



Hình 3.2 Quá trình huấn luyện mạng nơ ron

Kết quả cho thấy mạng dự đoán với tỉ lệ đúng 98% cho phân loại kết quả tín dụng.

1. **Kết quả chương trình**

Kết quả thử nghiệm mô hình dự đoán lỗi từ mạng nơ ron trên phần mềm duyệt tín dụng được thể hiện dưới đây. Như đã trình bày trước đó, hai ngưỡng trên và ngưỡng dưới (trong đoạn [0;1]) khi so sánh kết quả mạng nơ ron và phần mềm kiểm thử được đặt ra để giới hạn kết luận dựa trên khoảng cách tuyệt đối, cụ thể là:

* Ngưỡng [0,0.1]: Kết luận không lỗi khi tương đồng hoàn toàn. Kết luận có lỗi khi có khác biệt rất nhỏ
* Ngưỡng [0;0.5]: Kết luận không lỗi khi tương đồng hoàn toàn. Kết luận có lỗi khi có khác biệt vừa phải
* Ngưỡng [0,2;0,8]: Kết luận không lỗi khi tương đồng nhỏ. Kết luận có lỗi khi khác biệt khá lớn

Kết quả thực nghiệm với các mức ngưỡng trên được thể hiện trong bảng dưới. Bảng bao gồm kết quả trên tất cả các loại lỗi được đưa vào phần mềm (Bảng 5), tỉ lệ đoán đúng lỗi, đoán nhầm lỗi và không kết luận được là lỗi hay không. Ngưỡng được chọn là ngưỡng cho tỉ lệ lỗi trung bình tốt nhất.



Bảng 3.4 Tỉ lệ lỗi của mô hình dự đoán

1. **Tổng kết chương**

# Kết luận

1. **Kết luận**
2. **Định hướng phát triển**

# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | W. E. Lewis, Software testing and continuous quality improvement,, CRC press, 2016. |
| [2] | P. C. Jorgense, Software Testing: A Craftsman’s Approach, Boca Raton, FL, USA, 1995: CRCPress, Inc., 1995. |
| [3] | A. P. O. G. B. C. F. M. Y. Zhauniarovich, "Towards black box testing of android app," in *2015 10th International Conference on Availability, Reliability and Security (ARES)*, Toulouse, France, 2015. |
| [4] | M. K. N. L. L. J. M. V. P. A. Laplante, "Building caring healthcaresystems in the internet of things,," *EEE Systems Journal ,* 2017. |
| [5] | K. Cường, "Tỷ lệ kỹ sư kiểm thử phần mềm ở VN còn thấp," 15 7 2011. [Online]. Available: https://vnexpress.net/so-hoa/ty-le-ky-su-kiem-thu-phan-mem-o-vn-con-thap-1513971.html. |
| [6] | D. Dang, "Open Source Test Automation: Riding the Second Wave," in *STAREAST 2016 - Software Testing Conference*, 2016. |
| [7] | "AI in Testing: The third wave of automation," CloudQA, [Online]. Available: https://cloudqa.io/ai-in-testing-the-third-wave-of-automation/. [Accessed 2019]. |
| [8] | G. G. A. P. A. Z. M. Beller, "When, how, and why developers (do not) test in their ides," in *10th Joint Meeting on Foundations of Software Engineering*, 2015. |
| [9] | Q. T. M. Godfrey, "Growth, evolution, and structural change in open source software," in *4th international workshop on principles of softwareevolution, ACM*, 2001. |
| [10] | N. K. F. H. S. A. I. S. Basri, " Predicting effort for requirementchanges during software developmen," in *Seventh Symposiumon Information and Communication Technology, ACM,* , 2016. |
| [11] | J. M. B. A. B.-H. U. Kanewala, "Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels," *Software: Testing, Verification and Reliability ,* vol. 26, no. 3, pp. 245-269, 2016. |
| [12] | C. Kaner, "Exploratory testing," in *Quality Assurance Institute Worldwide Annual Soft-ware Testing Conference*, Orlando, FL, 2006. |
| [13] | W. W. Royce, "Managing the development of large software systems,," *Proceedings of IEEE WESCON,* no. 26, pp. 1-9, 1970. |
| [14] | J. a. G. M. Voas, Software Fault Injection, Hoboken, NJ: John Wiley & Sons, 1998. |
| [15] | M. N. Dhingra, " Contingent study of black box and white box testing tech-niques," *International Journal of Current Engineering and Technology,* 2014. |
| [16] | A. Kolawa and D. Huizinga, Automated Defect Prevention: Best Practices in Software Management, Wiley-IEEE Computer Society Press, 2007. |
| [17] | K. Beck, Test-Driven Development by Example, Vaseem: Addison Wesley, 2002. |
| [18] | D. Graham, Foundations of Software Testing: ISTQB Certification, Int. Thomson Business Press, 2006. |
| [19] | I. C. B. M. M. H. A. Leitner, "Reconciling manual and automatedtesting: The autotest experience,," in *40th AnnualHawaii International Conference*, 2007. |
| [20] | Microsoft Corporation, " Designing a Test Environment," 2009. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc778654(v=ws.10). |
| [21] | M. Fowler, "TestCoverage," 17 4 2012. [Online]. Available: https://martinfowler.com/bliki/TestCoverage.html. |
| [22] | R. Ramler, "Economic Perspectives in Test Automation: Balancing Automated and Manual Testing with Opportunity Cost.," in *International Workshop on Automation of Software Test,*, Shanghai, China, 2006. |
| [23] | S. Berner, R. Weber and R. Keller, "Observations and lessons learned from automated testing," in *IEEE*, Saint Louis, MO, USA, USA, 2005. |